

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2000-76118

(P2000-76118A)

(43)公開日 平成12年3月14日(2000.3.14)

(51)Int.Cl.

G 0 6 F 12/00

識別記号

5 4 5

5 4 6

F I

G 0 6 F 12/00

テーマコード(参考)

5 4 5 B

5 B 0 8 2

5 4 5 M

5 4 6 K

審査請求 未請求 請求項の数16 O L (全 40 頁)

(21)出願番号

特願平10-249235

(22)出願日

平成10年9月3日(1998.9.3)

(71)出願人 000005496

富士ゼロックス株式会社

東京都港区赤坂二丁目17番22号

(72)発明者 齋藤 淳

神奈川県足柄上郡中井町境430 グリーン

テクなかい 富士ゼロックス株式会社内

(72)発明者 川邊 恵久

神奈川県足柄上郡中井町境430 グリーン

テクなかい 富士ゼロックス株式会社内

(74)代理人 100086531

弁理士 澤田 俊夫

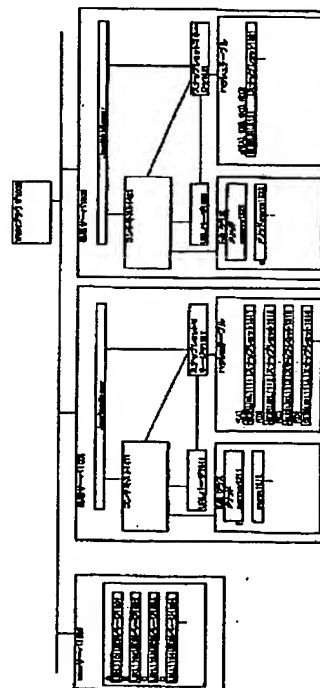
Fターム(参考) 5B082 HA03 HA05

(54)【発明の名称】 分散ファイル処理装置および分散ファイル処理方法

(57)【要約】

【課題】 複製を有効に利用してファイルサーバへの負荷や、経路上の通信トラフィックを軽減した分散ファイル処理装置および方法を提供する。

【解決手段】 原料を編集加工する手続きを表す手続き名等で修飾して構成した仮想URLに基づいて、原料ファイルの処理を行うコンテキスト手段と、コンテキスト手段において処理を行った原料ファイルの処理結果を保持する結果保持手段を有し、さらに、結果保持手段の保持する処理結果の有効性を判定する結果管理手段とを有する。結果管理手段は、処理結果に対応する原料ファイル名、手続き名、手続き処理の処理実行日時、手続き処理実行を要求した要求固有の処理要求識別子等に基づいて処理結果の有効性を判定する。



【特許請求の範囲】

【請求項 1】 コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理装置において、

前記ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファイル名に対応する原料ファイルのデータを入力し、前記取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段と、

前記コンテキスト手段において起動した前記手続きによって処理を行った原料ファイルの処理結果を保持する処理結果保持手段と、

前記処理結果保持手段の保持する前記処理結果を取り出して出力する結果出力手段と、

前記処理結果保持手段の保持する処理結果の有効性を判定する処理結果管理手段と、

を有することを特徴とする分散ファイル処理装置。

【請求項 2】 前記処理結果管理手段は、前記処理結果保持手段の保持する処理結果に対応する原料ファイル名およびコンテキスト手段において実行された手続き名に基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項 1 記載の分散ファイル処理装置。

【請求項 3】 前記処理結果管理手段は、前記コンテキスト手段による手続き処理の処理実行日時、

前記コンテキスト手段による手続き処理実行を要求した要求固有の処理要求識別子、の少なくともいずれかに基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項 2 記載の分散ファイル処理装置。

【請求項 4】 前記処理結果管理手段は、前記原料ファイルの更新日時と、前記コンテキスト手段による手続き処理の処理実行日時とを比較し、該比較結果に基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項 2 記載の分散ファイル処理装置。

【請求項 5】 前記コンテキスト手段が、前記修飾ファイル名の解釈において、要求処理結果の処理時刻指定子が含まれると判定した場合は、

前記処理結果管理手段は、前記時刻指定子の示す時間範囲に前記処理日時が対応するか否かに基づいて、前記処理日時が対応づけられた処理結果の有効性を判定する構成を有することを特徴とする請求項 3 記載の分散ファイル処理装置。

【請求項 6】 前記ネットワークは、

複数のコンテキスト手段、該複数のコンテキスト手段に対応して設けられた複数の処理結果保持手段、および該複数の処理結果保持手段に対応して設けられた複数の処理結果管理手段を接続した構成を有し、

前記複数のコンテキスト手段は、それぞれ独立に手続き処理が可能であり、

ネットワーク上に出力された処理要求に対応する処理結果の有効性を判定する第 1 の処理結果管理手段は、

10 該処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の第 2 の処理結果保持手段を管理する第 2 の処理結果管理手段に処理結果の有効性についての問い合わせを実行する構成を有することを特徴とする請求項 1 記載の分散ファイル処理装置。

【請求項 7】 前記コンテキスト手段における修飾ファイル名の解釈において、前記処理結果保持手段に保持された処理結果の無効化または破棄いずれかの指示を定めた所定のパラメータが含まれることが解析されたことを条件として、

20 前記処理結果管理手段は、前記処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理を実行する構成を有することを特徴とする請求項 1 記載の分散ファイル処理装置。

【請求項 8】 前記処理結果管理手段による前記処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理の実行に伴い、

30 前記コンテキスト手段は、前記無効化または破棄を行った処理結果に対応する原料ファイルを取得して、新たに処理を実行して前記処理結果保持手段に保持する構成を有することを特徴とする請求項 7 記載の分散ファイル処理装置。

【請求項 9】 前記コンテキスト手段における原料ファイルに対する前記手続きによる処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて前記処理結果保持手段に保持する構成としたことを特徴とする請求項 1 に記載の分散ファイル処理装置。

【請求項 10】 前記コンテキスト手段は、前記修飾ファイル名の解釈において、処理の要求が非同期的であることを示す指定データが含まれる判定がなされたことを条件として、原料ファイルに対する前記手続きを処理要求タイミングとは独立に非同期的処理として実行する構成を有することを特徴とする請求項 9 に記載の分散ファイル処理装置。

【請求項 11】 前記コンテキスト手段が前記修飾ファイル名の解釈において、処理の要求条件に完全一致しないが、部分一致する処理結果が前記処理結果保持手段に

含まれると判定した場合において、前記結果出力手段は、前記処理結果保持手段に含まれる部分一致する処理結果を出力する構成を有することを特徴とする請求項 1 に記載の分散ファイル処理装置。

【請求項 1 2】 コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理方法において、

前記ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファイル名に対応する原料ファイルのデータを入力し、前記取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段による手続き処理ステップと、

前記手続き処理ステップにおいて起動した前記手続きによって処理を行った原料ファイルの処理結果を処理結果保持手段に保持する結果保持ステップと、前記処理結果保持手段の保持する処理結果の有効性を判定する処理結果有効性判定ステップと、前記処理結果有効性判定ステップにおいて有効と判定された処理結果であり、前記処理結果保持手段に保持された処理結果を取り出して出力する結果出力ステップと、を有することを特徴とする分散ファイル処理方法。

【請求項 1 3】 前記処理結果有効性判定ステップは、前記処理結果保持手段の保持する処理結果に対応する前記原料ファイル名および前記コンテキスト手段において実行された手続き名に基づいて前記処理結果の有効性を判定するステップを含むことを特徴とする請求項 1 2 記載の分散ファイル処理方法。

【請求項 1 4】 前記処理結果有効性判定ステップは、前記コンテキスト手段による手続き処理の処理実行日時、前記コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれかに基づいて前記処理結果の有効性を判定するステップを含むことを特徴とする請求項 1 3 記載の分散ファイル処理方法。

【請求項 1 5】 前記処理結果有効性判定ステップは、前記原料ファイルの更新日時と、前記コンテキスト手段による手続き処理の処理実行日時を示す処理日時とを比較し、該比較結果に基づいて処理結果の有効性判定を行うステップを含むことを特徴とする請求項 1 3 記載の分散ファイル処理方法。

【請求項 1 6】 前記コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続き

によって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて前記処理結果保持手段に保持することを特徴とする請求項 1 2 に記載の分散ファイル処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は分散ファイル処理装置および処理方法に関する。さらに詳細には複数のコンピュータが接続されたデータ通信ネットワークを介してネットワーク上のサーバー等に分散して存在するデータに対して効率のよいアクセスを可能とし、迅速な処理データの抽出を可能とした分散ファイル処理装置及び分散ファイル処理方法に関する。

【0002】

【従来の技術】 ネットワークを介する分散ファイルアクセスシステムにおいては昨今、ますますその迅速、正確性に対する要求が高まってきている。従来から、様々な分散ファイル処理システムが提案され、また実行されているが、その代表的なものを以下に説明する。

【0003】 [従来技術 1] 従来の分散ファイルアクセスシステムにおける一つの一般的方式として、ファイルを提供するファイルサーバとファイルをアクセスするクライアントのネットワークの経路上に、代理サーバや中継サーバ（以降代理サーバ）を分散的に配置して、クライアントはこれを経由してファイルにアクセスする方式がある。例えば、FTP ミラーサーバなどがこれに相当する。

【0004】 このような代理サーバを用いたシステムでは、あらかじめファイルの複製を提供することで、ファイルサーバへの負荷や、ネットワーク経路上の通信トラフィックを軽減できるというメリットがある。

【0005】 また、あらかじめファイルの複製をサーバに用意しておくのではなく、クライアントからアクセスされたファイルに対して動的に複製を作成し、その後のアクセスに提供するスナップショット方式も考案されている。例えば、RFC 2187 で述べられている HTTP 中継サーバプログラム Squid (<http://squid.nlanr.net/Squid/>において配布されている) においては、複製の更新や無効化は、複製の生成時刻、オリジナルのファイルの生成時刻、などのタイムスタンプ情報に基づく計算により必要と判定されたときに行われる。

【0006】 このようなスナップショット方式では、例えば、次のような手順でファイル複製の管理が行われる。クライアントからのファイルアクセス要求が発生し、要求されたファイルの複製が代理サーバにあった場合に、前記タイムスタンプ情報に基づく計算を行い、複製の新しさを判定する。複製が十分新しい場合以外は、要求されたファイルのオリジナルのタイムスタンプと複製のタイムスタンプとを比較して、複製の方が新しい場

合は複製を返す。複製の方が古い場合は、古いと判定された複製を無効化し、新たにオリジナルファイルをコピーして、複製を作成し返す。このような動的に複製を作成する方式では、あらかじめ複製を用意する方式に比べ、複製の管理が自動化できるので管理コストが軽減され、代理サーバの増設も容易である。

【0007】ところが、WWWのCGIで代表されるような、ファイルサーバ上でアクセス時にオンデマンド型で生成されるファイル（例えば、今日の天気 をファイル内の情報に含む場合）では、タイムスタンプ情報は、アクセス時に付与される。タイムスタンプ情報のみを用いて複製を管理する方式では、複製とオリジナルの比較を行うための適当なタイムスタンプを決定する機構がないため、複製が有効であるかどうかの判定ができない。そのため、動的に生成されるファイルに関して複製を有効に利用することは困難である。この種の技術を従来技術1とする。

【0008】〔従来技術2〕特開平8-292910には、原料となる資源名から、それを加工する手続きを表す手続き名と、手続きに対するパラメータと、手続き名を解釈し手続きを起動する装置をあらわすコンテキスト識別子とで修飾して構成した資源名を解釈して、原料資源へのアクセスを行い、手続きを起動して、原料資源の処理を行う資源管理装置および資源管理方式が開示されている。

【0009】この特開平8-292910に開示された資源管理装置および資源管理方式では、資源名に埋め込まれた手続き名や原料資源名が、分散的に解決され、手続き同士がネットワーク上でパイプライン的に結合される。しかし、手続きが多段に結合されたり、複数の入力を持つ手続きの場合はパイプラインがツリー構造のように構成されることを考えると、ネットワーク上のバースト的な通信エラーでいずれか一個所の通信が失敗した場合、最終的な処理結果である合成資源が得られないという欠点がある。通信に失敗して結果が得られなかった場合は、再実行を行って結果を得ることもできるが、一つの合成資源を得るのに複数のネットワーク通信が必要なので、信頼性の低いネットワークでは、いずれかの通信が高い確率で失敗し、結果を得るまでに何度も再実行を繰り返す必要があったり、合成資源が得られないこともある。この種の技術を従来技術2とする。

【0010】

【発明が解決しようとする課題】分散ファイル処理において、作成された複製を有効に利用してファイルサーバへの負荷や、ネットワーク経路上の通信トラフィックを軽減することを目標とすることを考慮すると、上述した従来技術の構成では以下に説明する問題点が発生する。

【0011】まず、従来技術1は、上述のようにWWWのアクセスにおいて、複製を利用して通信トラフィックを減らすなどの効果は提供するが、クライアント等から

のリクエストを受けたときにサーバ上で動的に生成される文書に関しては複製を取り扱うのが困難となる。

【0012】また、従来技術2は分散ファイルアクセスの手順を用いて分散ファイル処理を実現でき、各種の処理を組み合わせたサービスを提供することが可能となるが、処理結果がパイプラインで複数組み合わせることを必要とするため、信頼性の低いネットワークにおいて高い確率で失敗が起きやすく、実用性が低いという問題点を有する。

10 【0013】本発明はこのような従来技術の有する各種の問題点を解決するものであり、複製を有効に利用してファイルサーバへの負荷や、経路上の通信トラフィックを軽減することを達成することを目的とする。

【0014】

【課題を解決するための手段】本発明は上記の目的を達成するものであり、コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理装置において、ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファイル名に対応する原料ファイルのデータを入力し、取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段と、コンテキスト手段において起動した手続きによって処理を行った原料ファイルの処理結果を保持する処理結果保持手段と、処理結果保持手段の保持する処理結果を取り出して出力する結果出力手段と、結果保持手段の保持する処理結果の有効性を判定する処理結果管理手段とを有することを特徴とする。

【0015】さらに、本発明の分散ファイル処理装置において、管理手段は、処理結果保持手段の保持する処理結果に対応し原料ファイル名およびコンテキスト手段における手続き名に基づいて処理結果の有効性を判定する構成を有することを特徴とする。

40 【0016】さらに、本発明の分散ファイル処理装置において、処理結果管理手段は、コンテキスト手段による手続き処理の処理実行日時を示す処理日時、コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれかに基づいて処理結果の有効性を判定する構成を有することを特徴とする。

50 【0017】さらに、本発明の分散ファイル処理装置において、処理結果管理手段は、原料ファイルの更新日時と、コンテキスト手段による手続き処理の処理実行日時を示す処理日時とを比較し、該比較結果に基づいて処理結果の有効性を判定する構成を有することを特徴とする。

【0018】さらに、本発明の分散ファイル処理装置は、コンテキスト手段が、修飾ファイル名の解釈において、要求処理結果の処理時刻指定子が含まれると判定した場合は、処理結果管理手段は、時刻指定子の示す時間範囲に処理日時が対応するか否かに基づいて、処理日時が対応づけられた処理結果の有効性を判定する構成を有することを特徴とする。

【0019】さらに、本発明の分散ファイル処理装置において、ネットワークは、複数のコンテキスト手段、該複数のコンテキスト手段に対応して設けられた複数の処理結果保持手段、および該複数の処理結果保持手段に対応して設けられた複数の処理結果管理手段を接続した構成を有し、複数のコンテキスト手段は、それぞれ独立に手続き処理が可能であり、ネットワーク上に出力された処理要求に対応する処理結果の有効性を判定する第1の処理結果管理手段は、該処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の第2の処理結果保持手段を管理する第2の処理結果管理手段に処理結果の有効性についての問い合わせを実行する構成を有することを特徴とする。

【0020】さらに、本発明の分散ファイル処理装置は、コンテキスト手段における修飾ファイル名の解釈において、処理結果保持手段に保持された処理結果の無効化または破棄いずれかの指示を定めた所定のパラメータが含まれることが解析されたことを条件として、処理結果管理手段は、処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理を実行する構成を有することを特徴とする。

【0021】さらに、本発明の分散ファイル処理装置は、処理結果管理手段による処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理の実行に伴い、コンテキスト手段は、無効化または破棄を行った処理結果に対応する原料ファイルを取得して、新たに処理を実行して処理結果保持手段に保持する構成を有することを特徴とする。

【0022】さらに、本発明の分散ファイル処理装置において、コンテキスト手段における原料ファイルに対する手続きによる処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて処理結果保持手段に保持する構成としたことを特徴とする。

【0023】さらに、本発明の分散ファイル処理装置において、コンテキスト手段は、修飾ファイル名の解釈において、処理の要求が非同期的であることを示す指定データが含まれる判定がなされたことを条件として、原料ファイルに対する手続きを処理要求タイミングとは独立に非同期的処理として実行する構成を有することを特徴とする。

【0024】さらに、本発明の分散ファイル処理装置は、コンテキスト手段が修飾ファイル名の解釈において、処理の要求条件に完全一致しないが、部分一致する処理結果が処理結果保持手段に含まれると判定した場合において、結果出力手段は、処理結果保持手段に含まれる部分一致する処理結果を出力する構成を有することを特徴とする。

【0025】さらに、本発明の分散ファイル処理方法は、コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理方法において、ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファイル名に対応する原料ファイルのデータを入力し、取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段による手続き処理ステップと、手続き処理ステップにおいて起動した手続きによって処理を行った原料ファイルの処理結果を結果保持手段に保持する結果保持ステップと、処理結果保持手段の保持する処理結果の有効性を判定する処理結果有効性判定ステップと、処理結果有効性判定ステップにおいて有効と判定された処理結果であり、処理結果保持手段に保持された処理結果を取り出して出力する結果出力ステップとを有することを特徴とする。

【0026】さらに、本発明の分散ファイル処理方法において、処理結果有効性判定ステップは、処理結果保持手段の保持する処理結果に対応する原料ファイル名およびコンテキスト手段における手続き名に基づいて処理結果の有効性を判定することを特徴とする。

【0027】さらに、本発明の分散ファイル処理方法は、コンテキスト手段による手続き処理の処理実行日時、コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれか一方に基づいて処理結果の有効性を判定することを特徴とする。

【0028】さらに、本発明の分散ファイル処理方法において、処理結果有効性判定ステップは、原料ファイルの更新日時と、コンテキスト手段による手続き処理の処理実行日時とを比較し、該比較結果に基づいて処理結果の有効性判定を行うことを特徴とする。

【0029】さらに、本発明の分散ファイル処理方法において、コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別

子を生じ、該処理結果識別子を処理結果に対応させて処理結果保持手段に保持することを特徴とする。

【0030】

【発明の実施の形態】本発明の分散ファイル処理システムを構成するネットワーク上の主要な要素について、その概要を図1に示す。図1に示すようにネットワーク上には、ファイルアクセスの実行主体となるクライアント（図1右端）が接続され、また、ファイルの供給主体であり原料ページA、B、C...を保持するサーバ（図1左端）が接続されている。さらに本発明のシステムでは、サーバの保有する原料ページA、B、C...をさまざまな形で加工し、またその加工結果を保持する等、各種の機能を有する処理サーバが存在し、これらが図1ではコンテキスト手段、処理結果保持手段等として示されている。これらの処理サーバの具体的な構成、機能については以下の実施例において詳細に説明する。

【0031】図1に示す本発明の基本構成における処理の概要について簡単に説明する。まず、ネットワーク上のユーザがWWWクライアントを介してある特定ファイルの加工データを要求する。要求されたファイルの原料ファイルは基本的にWWWサーバの原料ページA、B、C...として保持されているものである。ユーザはこれらの原料ファイルに対して様々な態様で加工した結果データの出力を要求する。例えば原料ページAに対応するHTML文書の一部を抽出して一つの要約されたHTML文書として出力（メソッドsummary）することや、複数のHTML文書を合成して一つのHTML文書として出力（メソッドmerge）すること等である。これらの加工処理が図1に示す手続きf、g、h...の一つの態様である。

【0032】図1に示す処理結果保持手段が保持するf(A)、f(B)等は原料ページA、Bに対して、所定の処理を施して得られた結果データである。処理結果保持手段は、WWWクライアントからの要求に応じた処理を行った処理データを保持している場合と保持していない場合がある。また、加工データ、例えばf(A)自体は有していても、ユーザのアクセス時には、既にその加工源となった原料ページAが更新され更新原料ページA'として、WWWサーバに保持されている場合もある。このような、様々な場合にWWWクライアントからの要求に柔軟に対応できるシステムを実現するのが本発明のシステムであり、その詳細を以下の実施例において説明する。

【0033】【実施例1】まず、本発明の分散ファイル処理装置における第1の実施例を説明する。実施例1の分散ファイル処理装置の構成ブロック図を図2に示す。

【0034】図2に示すように、ネットワークには、原料ファイルをもつWWWサーバ1000と、コンテキストおよび各種の手続きを実行する処理サーバ1001、1002と、ファイルリクエストを出力するWWWブラウザ1003が接続されている。なお、図2では処理サーバがネットワークに2つ接続されている例を示しているが、これら処理サーバ、WWWサーバ、WWWブラウザの接続数は任意である。

【0035】WWWサーバ1000上に原料ファイル1801があり、そのURLは例えば、
http://host0/page01.html
であり、これをURL1101とする。WWWサーバ1000は、さらに他の原料ページ1802、1803、1804...を有し、これら原料ページ各々に対してURL1102、URL1103、URL1104...が対応する。

【0036】処理サーバ1001、1002（JWS：Java Web Server）上にJavaのクラスであるEditがあり、メソッドとして、Editクラスのメソッドであるメソッドsummary1211、1221、merge1212、1222他が存在する。

【0037】この実施例では、図2に示すようにファイルサーバはWWWサーバ1000で、ファイルはHTMLフォーマットで記述されたWWWページである。WWWサーバ1000と、そのクライアントであるWWWブラウザ1003は、HTTP（HyperText Transfer Protocol）でネットワークを介して、WWWページを送受信する。ファイル名はURLで、例えば、
http://host1/index.html
などと記される。URLのシンタクスは、RFC1738で規定されている。

【0038】例えば、処理サーバ1001のホスト名が、host1であり、ポート番号は8080であるとすると、WWWブラウザ1003は、原料ファイル名URL1101を手続き名、パラメータ、コンピュータ名等で修飾した次のような仮想URL1111によりリクエストを送出することができる。

http://host1:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.html

仮想URLについては、以下において詳細に説明する。

【0039】以下、実施例1について、

「1. 仮想URL」

「2. 各構成要素の概要」（図2参照）

「3. 処理サーバの動作」（図3参照）

「4. コンテキストの動作」（図4参照）

「5. URLパーザの動作」（図5参照）

「6. スナップショットマネージャの動作」（図6、図7参照）

「7. レスポンス時の処理」

の順で詳細に説明する。

【0040】「1. 仮想URL」まず、仮想URLについて詳細に説明する。原料ファイル名を手続き名、パラメータ、コンピュータ名で修飾したファイル名は、「URL」を手続き名、パラメータ、コンピュータ名で修飾した「仮想URL」として実現する。仮想URLの記述を、例をあげて説明する。仮想URLは、例えば次のように記述される。

```
http://host1:8080/context/
Edit?method=summary&url1=
http%3A%2F%2Fhost0%2Fpage01.html
```

【0041】仮想URLの先頭部には、ファイルアクセスのプロトコル、コンテキストの存在する処理サーバのホスト名、TCPポート番号、コンテキスト起動を指示するパス名が記述され、通常のURLと同様のシンタクスである。

【0042】例えば、プロトコルがHTTPで、ホスト名host1、ポート番号8080、パス名がcontext/であるとすれば、仮想URLの先頭部は、
http://host1:8080/context/ 20
であって、それに続き、コンテキストおよび手続きが指定される。

【0043】コンテキストはこの実施例ではJava言語で記述されたオブジェクトプログラムとして実現され、クラス名で指定する。手続きはJavaで記述されたメソッドで、前記クラス名とメソッド名で指定される。メソッド名は、次の形式で仮想URL中に含める。先頭が文字 ? で始まり、<propertyName>=<propertyValue>の形をなす文字列を文字 & で区切って連結する形式であり、propertyNameは、propertyValueとして記述された値の種類を識別するための名前である。メソッド名は、method= に続く文字列で指定される。例えば、クラス名Edit、メソッド名summaryならば、Edit?method=summaryである。

【0044】パラメータ、原料のURLも、<propertyName>=<propertyValue>の形式で記述され、& を区切りとしてメソッド名の後ろに追加される。また、この形式ではメソッド名やパラメータや原料のURL中の文字 / , & , ? , : , = はそれぞれ %2F , %26 , %3F , %3A , %3D に変換される。

【0045】例えば、原料ファイルのURLが、http://host0/page01.htmlで示されるとき、これを仮想URL中に記述すると、&url1=http%3A%2F%2Fhost0%2Fpage01.htmlに変換される。

【0046】このように記述される仮想URLを解析す 50

るURLパーザ1611、1621の動作の詳細については、図5を用いて後段で説明する。

【0047】なお、以下の説明においては、冗長性を回避するため仮想URL等の記述を略記法によって簡略化し記号に置き換えて説明する。仮想URL等と各記号を以下のように対応づける。

【0048】まず、サーバ名、ポート番号、コンテキスト、クラス名、メソッド名をまとめて、英小文字の記号で表現する。

f: host1上のポート8080番のWWWサーバによりアクセスできるコンテキストが起動するクラスEditのメソッド「summary」を「f」とする。

g: host2上のポート8080番のWWWサーバによりアクセスできるコンテキストが起動するクラスEditのメソッド「merge」を「g」とする。

【0049】さらに、URL（仮想URLも含む）を英大文字の記号で表現する。例えば、

A: 「http://host0/page01.html」を「A」とする。

B: 「http://host0/page02.html」を「B」とする。

X: 「http://host1:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.html」を「X」とする。

Y: 「http://host1:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage02.html」を「Y」とする。

Z: 「http://host2:8080/context/Edit?method=merge&url1=http%3A%2F%2Fhost1%3A8080%2Fcontext%2FEdit%3Fmethod%3Dsummary%26url1%3Dhttp%253A%252F%252Fhost0%252Fpage01.html&url2=http%3A%2F%2Fhost1%3A8080%2Fcontext%2FEdit%3Fmethod%3Dsummary%26url1%3Dhttp%253A%252F%252Fhost0%252Fpage02.html」を「Z」とする。

【0050】また、手続きに対する原料としてURLを指示する記述、すなわちURLが修飾されて仮想URLを構成する構造を、関数の表現法と同様に、以下のように表す。例えば、前記A、B、X、Y、Z、f、gを用いて、Xをf(A)、Yをf(B)、Zはg(X、Y)、または、g(f(A)、f(B))と表す。

【0051】「2. 各構成要素の概要」次に本実施例を構成する処理サーバの各要素について、図1と図2を参照しながらその概要を説明する。図1で示すコンテキス

ト手段は、実施例1では次のようなコンテキストオブジェクト（以下、混乱のないかぎりコンテキストと記す）である。図2で示すコンテキスト1411、1421は、WWWサーバがHTTPによるファイルの読みだしリクエストとしてGETや、書込みのリクエストとしてPUTをWWWブラウザ1003から受けとった時に、WWWサーバから起動される外部プログラムであって、図4で示す動作フローに従って動作する。コンテキストの詳細な動作は図4を用いて後段で詳細に説明する。

【0052】WWWサーバが外部プログラムを起動する方法は、CGI方式やServlet方式（Servlet方式についてはJavaWebServer version1.1中の文書 The Java Servlet API に記されている。）がよく知られている。この実施例では、WWWサーバは、Sun Microsystems, Inc. のJavaWebServer（以降JWSとする）とし、コンテキストはJWSが起動する外部プログラムとして、Servletの形式で実装されている。

【0053】図1で示す結果保持手段は、実施例1では図2で示すようにJavaのハッシュテーブルとして実現され、処理を行なって生成された処理結果のオブジェクトをスナップショットオブジェクトとして、仮想URLをキーとして取り出せる状態で保持する。図2で示す処理サーバ1001は、スナップショット1311、1312、1313、1314をハッシュテーブルに保持し、それぞれ仮想URL1111、1112、1113、1114をキーとして取り出すことができる。また、処理サーバ1002は、スナップショット1321を保持し、仮想URL2121をキーとして取り出すことができる。結果出力手段は処理結果のオブジェクトを、JWSを介してHTTPプロトコルを用いて出力するプログラムとして実現される。

【0054】図1で示す結果管理手段は、本実施例では、図2で示すようにスナップショットを保持したハッシュテーブルを管理するServlet形式のプログラムとして実現され、スナップショットマネージャ1511、1521と呼ばれる。スナップショットマネージャ1511、1521には、スナップショットの有効性を判定するメソッドが実装されている。有効性を判定するメソッドには、他のJavaプログラムから呼び出され、Javaの論理値型の値を返すものと、JWSがリクエストに応じて起動して、スナップショットが有効かどうかの結果をHTTPヘッダ中に出力するものがある。スナップショットマネージャの詳細な動作は図6、図7を用いて後段で説明する。

【0055】「3. 処理サーバの動作」次に、処理サーバの動作の詳細について説明する。サーバの動作フローを図3に示す。例として、WWWブラウザ1003に、`http://host1:8080/context`

`/Edit?method=merge&url=http%3A%2F%2Fhost0%2Fpage01.html`

というURLをセットして、ファイルを読み込む指示を出力した場合について説明する。

【0056】WWWブラウザ1003からの上記のファイル読み込み指示出力に対応して、図3のステップ2010においてhost1というDNS名を有するホスト名のJWS、例えば図2で示す処理サーバ1001は次のようなHTTPのリクエストを受けとる。

`GET/context/Edit?method=merge&url=http%3A%2F%2Fhost0%2Fpage01.html HTTP/1.0`

【0057】ステップ2020においてJWSは、Context.class（以降、contextとする）というJavaのサーブレット形式のプログラムが初期化されているかどうか調べる。もし初期化されていなければ、ステップ2030においてクラスファイルを読み込んでメモリにロードして初期化し、実行可能な状態とした後、実行を開始する。

【0058】ステップ2040においてJWSはサーブレットcontextのメソッドserviceを引数のオブジェクトHttpRequest, HttpServletResponseを渡して起動する。

【0059】ステップ2050において、コンテキストは、ファイルに対する処理を行い、処理結果をスナップショットオブジェクトとして保持する。引数のオブジェクトHttpRequestから、クライアントであるWWWブラウザへの出力ストリームを保持するServletOutputStreamを取り出し処理結果を書き込む。ステップ2050におけるコンテキストの詳細な動作は図4に示すとおりであり後述する。

【0060】ステップ2060においてHTTPリクエストのコネクションに対し、ServletOutputStreamに書き込まれた内容が出力される。

【0061】「4. コンテキストの動作」前記図3のステップ2050におけるコンテキストの動作を図4に示すコンテキストの動作フローにより詳細に説明する。

【0062】図4におけるステップ3010においてサーブレットcontextのメソッドserviceが引数のオブジェクトHttpRequest, HttpServletResponseを受けとって起動する。

【0063】ステップ3020においてオブジェクトHttpRequestのgetRequestUriメソッドにより、仮想URLに含まれる文字列、

`/context/Edit?method=merg`

e&url1=http%3A%2F%2Fhost0%2Fpage01.html

を取り出す。この文字列を先頭部と合成して次の仮想URLを再現する。

http://host1:8080/context/Edit?method=merge&url1=http%3A%2F%2Fhost0%2Fpage01.html

【0064】ステップ3030において仮想URLをURLパーザに渡して解析する。URLパーザの動作フローは図5に示す通りであり、この動作については後段で詳細に説明する。URLパーザはVURLクラスのオブジェクトを生成し、URL中に含まれる、メソッド名、原料のURL、パラメータをVURLオブジェクトのインスタンス変数にセットして返す。VURLオブジェクトは、仮想URLの先頭部、クラス名、メソッド名、パラメータ、原料のURL、をインスタンス変数として持つオブジェクトである。

【0065】ステップ3040においてコンテキスト1411はスナップショットマネージャ1511にVURLオブジェクトを渡し、仮想URLに対応するスナップショットが有効であるかどうかの答を得る。スナップショットマネージャ1511の動作は図6、図7に示すとおりであり、後述する。

【0066】仮想URLに対応する有効なスナップショットが無かったときは、ステップ3050において、VURLオブジェクトのインスタンス変数に格納されている原料のURLに対してHTTPリクエストを送出し、原料のファイルを取得する。ステップ3060においてすべての原料ファイルを取得したかどうかを調べ、すべて取得していなければステップ3050に戻る。

【0067】ステップ3070において、ステップ3050で取得したファイルを引数として、VURLオブジェクトのインスタンス変数に格納されているクラス名とメソッド名で定まるメソッドを起動しその戻り値を処理結果としてオブジェクトを生成する。ステップ3080において処理結果のオブジェクトを仮想URLをキーにして前記ハッシュテーブルに格納する。ステップ3090において処理結果のオブジェクトをServletOutputStreamに書き込む。

【0068】ステップ3040において仮想URLに対応する有効なスナップショットがあった時は、ステップ3100においてそのスナップショットオブジェクトを仮想URLをキーとしてハッシュテーブルから取り出す。ステップ3110においてオブジェクトをServletOutputStreamに書き込む。

【0069】以下に、分散ファイルのプロトコルを用いて前記メソッドが起動され原料ファイルが処理され処理結果が仮想的にファイルとして得られる様子を説明する。前記メソッドはHTML文書の一部を抽出したHT

ML文書を作成するものとする。ただし、このメソッドはHTML文書を入力し、HTML文書を出力するプログラムであればどのようなものでも構わない。また、このメソッドの実現の詳細は本発明の動作とは無関係である。

【0070】図2に示す本実施例に関する全体ブロック図から理解されるように、原料ファイルをもつWWWサーバ1000と、コンテキストおよび手続きの動作する処理サーバ1001、1002と、WWWブラウザ1003がネットワークで接続されている。

【0071】WWWサーバ1000上に原料ファイル1801があり、そのURLがhttp://host0/page01.htmlであり、これをURL1101とする。

【0072】処理サーバ1001上にJavaのクラスであるEditがあり、前記メソッドはEditクラスのメソッドであるメソッドsummary1211とする。処理サーバ1001のホスト名が、host1であり、ポート番号は8080である。WWWブラウザ1003は、原料ファイル名URL1101を修飾した次のような仮想URL1111によりリクエストを送出する。

【0073】http://host1:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.html

【0074】ホスト名がhost1である処理サーバ1001が、WWWブラウザ1003からの仮想URL1111によるHTTPリクエストを受信すると、コンテキスト1411がJWSにより起動される。コンテキスト1411は、図4の手順にしたがって動作し、URLパーザ1611により仮想URL1111を解析して仮想URL1111を表すJavaオブジェクトVURL1111(obj)を生成する。

【0075】「5. URLパーザの動作」この仮想URL1111を入力した際のURLパーザ1611の動作を図5に示すフローを用いて説明する。URLパーザ1611は、図5におけるステップ4010において、仮想URL1111を受けとって起動する。ステップ4020において、仮想URL1111を文字 / を区切り文字として分解し、先頭部http://host1:8080/context/と残りの文字列Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.htmlに分ける。

【0076】ステップ4030において、残りの文字列から文字 ? を区切り文字として分解し、クラス名Editと残りの文字列method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.htmlに分ける。

【0077】ステップ4040において、残りの文字列から & を区切り文字として<propertyName>=<propertyValue>の形式の文字列を抽出する。メソッド名を含む文字列method=summaryが取り出される。

【0078】ステップ4050において、取り出された<propertyName>=<propertyValue>の形式の文字列について、propertyNameが method であるかどうかの判定をおこなう。メソッド名を含む文字列method=summaryについては、ステップ4060にすすむ。

【0079】ステップ4060において、メソッド名を含む文字列method=summaryから、propertyValueの値であるsummaryを取り出しメソッド名とする。

【0080】ステップ4080において、<propertyName>=<propertyValue>の形式の文字列の処理がすべて終わったかどうか調べ、終わっていないので、ステップ40に戻る。

【0081】ステップ4040において、残りの文字列から & を区切り文字として原料ページ名を含む文字列url1=http%3A%2F%2Fhost0%2Fpage01.htmlが取り出される。

【0082】ステップ4050において、propertyNameが method であるかどうかの判定をおこない、文字列url1=http%3A%2F%2Fhost0%2Fpage01.htmlについては、ステップ4070にすすむ。

【0083】ステップ4070において文字列url1=http%3A%2F%2Fhost0%2Fpage01.htmlから原料ページのURLであるhttp://host0/page1.htmlを取り出す。図2におけるURL1101である。

【0084】ステップ4080において、取り出した<propertyName>=<propertyValue>の形式の文字列の処理がすべて終わったのでステップ4090に進む。

【0085】ステップ4090において、仮想URLを表すVURLクラスのオブジェクトを生成し、先頭部http://host1:8080/context /、クラス名Edit、メソッド名summary、原料のURLであるhttp://host0/page01.htmlをインスタンス変数の値としてセットする。以上が仮想URL1111を入力した際のURLパーザ1611の動作である。

【0086】コンテキスト1411は、スナップショットマネージャ1511にオブジェクトVURL1111(obj)を渡し、仮想URL1111に対応するスナップショットオブジェクトの有効性を問い合わせる。

【0087】「6. スナップショットマネージャの動

作」スナップショットマネージャ1511は、図6、図7に示す動作フローの手順に従って、仮想URL1111に対応するスナップショットオブジェクトの有効性をしらべる。

【0088】スナップショットマネージャ1511が、他のスナップショットマネージャからのHTTPリクエストにより呼ばれるときは、JWSからServletとして起動され、ステップ5002から始まるメソッドserviceが呼ばれる。このHTTPリクエストは、有効性を調べるべきスナップショットに対応した仮想URLを、serviceメソッドの引数オブジェクトの中に含めるようなURLでなされる。例えば、サーバhost2（例えば図2の処理サーバ1002）のスナップショットマネージャ1521に、仮想URL、http://host2:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost0%2Fpage01.htmlに対応したスナップショットの有効性を問い合わせるリクエストに用いるURLは、

http://host2:8080/servlet/snapshot?url=http%3A%2F%2Fhost2%3A8080%2FEdit%3Fmethod%3Dsummary%26url1%3Dhttp%253A%252F%252Fhost0%252Fpage01.htmlである。

スナップショットマネージャが、contextサーバなどのJavaプログラムから呼ばれるときは、ステップ5010から始まるメソッドcheckLocalが起動される。

【0089】ステップ5002においてHttpServletRequest, HttpServletResponseオブジェクトを入力として起動する。ステップ5004において、HttpServletRequestオブジェクトのgetQueryString()メソッドを用いて、有効性を確認すべきスナップショットの仮想URLを取り出し、必要なら符号化を解く。ステップ5006において前記取り出した仮想URLをパーズしてVURLオブジェクトを生成する。ステップ5008においてVURLオブジェクトを引数としてステップ5010から始まるメソッドcheckLocalを起動する。

【0090】ステップ5010においてVURLオブジェクトVURL1111(obj)を入力として起動する。ステップ5020において、VURL1111(obj)が表す仮想URL1111に対応するスナップショットオブジェクトがあるかどうか調べる。これは、VURL1111(obj)のもつインスタンス変数の値から、仮想URL1111を構成し、仮想URL1111をキーとしてハッシュテーブルに保持されているスナ

ップショットオブジェクトがあるかどうかを調べる。あった時にはステップ5040にすすみ、なかった時にはステップ5030にすすむ。スナップショットがなかった場合にはステップ5030において、スナップショット無効を答えとして、Javaの論理値の偽として表現して出力し、呼び出し側に帰る。

【0091】スナップショットがあった場合にはステップ5040において、VURL1111 (obj) より原料のURLであるURL1101を取り出す。ステップ5050において取り出した原料のURLが、仮想URLではあるかどうかを調べ、仮想URLならステップ5060へ、仮想URLで無ければステップ5070へ進む。URL1101は仮想URLではないのでステップ5070に進む。

【0092】ステップ5060において、ステップ5050において取り出した原料のURLが仮想URLであった場合には、仮想URL中のホスト名で指定されるホストに対して、スナップショットマネージャに有効性を確認させるリクエストをHTTPで送る。リクエストに用いるURLは、JWSに対し、Servletであるスナップショットマネージャを起動させることを指示するものであり、前記原料のURLをserviceメソッドの引数オブジェクトの中に含めて渡すよう指示する。

【0093】そのリクエストに対し、レスポンスを返すスナップショットマネージャは、答えを、HTTPレスポンスヘッダにヘッダ名 Snapshot-Valid: に続けて出力し、レスポンスを送出する。レスポンスを受け取るスナップショットマネージャはそのヘッダから文字列 true または "false" を抽出して答えとする。

【0094】ステップ5070においてURL1101の原料ファイルの更新日時を取得する。ステップ5080においてURL1101のファイルの更新日時と仮想URL1111に対応するスナップショットオブジェクト1311の作成日時とを比較する。ファイルの更新日時が新しい場合は、論理値の答えを真とする。作成日時が新しい場合は答えを偽とする。

【0095】ステップ5090においてVURL1111 (obj) のインスタンス変数に格納された原料のURLすべてについてスナップショットが有効かどうかの判定が終了したかどうかを調べ、もし終了していなければステップ5040に戻る。ここではURL1101について結果を得ており、すべて終了したのでステップ5100に進む。ステップ5100において、ステップ5060または5080で得た、原料のURLについてのスナップショットの有効性の答えの論理積を計算する。ステップ5130において、ステップ5100において計算した論理積の値をスナップショットの有効性の判定結果の答えとして、Javaの論理値として表現して呼

び出し側に返す。

【0096】メソッドserviceのステップ5140において、メソッドcheckLocalの返回值としてJavaの論理値を受け取る。ステップ5150においてHttpServletResponseオブジェクトに対し、setHeaderメソッドを用い、HTTPレスポンスヘッダを表す変数にスナップショットの有効性の答えを格納する。答えは、ヘッダ名を表す文字列 Snapshot-Valid: に続き、メソッドcheckLocalの値が偽なら文字列 true を、値が偽なら文字列 false をヘッダの値として連結する。ステップ5160において、メソッドserviceを終了し、JWSが、HttpServletResponseの値をHTTPで出力する。

【0097】以上、図6、図7で示す手順がスナップショットマネージャ1511が、仮想URLに対応するスナップショットの有効性を答える手順である。

【0098】「7. レスポンス時の処理」

a. スナップショットが有効である場合、

「コンテキストの動作」の欄で説明したように、コンテキスト1411は、スナップショットが有効であると判定された場合、仮想URLに対応するスナップショットオブジェクト1311を取得する(図4、ステップ3100)。さらに、図3で説明された通りの手順でスナップショットオブジェクト1311がWWWブラウザに対するレスポンスとして出力(図3、ステップ2060)される。

【0099】b. スナップショットが無効である場合
コンテキスト1411は仮想URL1111に対応するスナップショットが無効であると判定された場合、原料ファイル名URL1101により、WWWサーバ1000に対しファイル取得のリクエストを送出する。このリクエストにより原料ファイル名URL1101に対応する原料ページ1801を取得(図4、ステップ3060)し、WWWブラウザ1003からのリクエストURL1111に指定されたメソッドsummary1211を起動する。メソッドsummary1211はHTML文書を入力とし、原料ページの中から特定の部分を抽出してそれを新規に作成したHTMLページ中に埋め込んだ処理結果のオブジェクトを出力する処理である。コンテキスト1411は出力されたオブジェクトをスナップショットオブジェクト1311として処理結果オブジェクトとして仮想URL1111とペアにしてハッシュテーブルに格納する(図4、ステップ3080)。その後、図3で説明された通りの手順でスナップショット1311がWWWブラウザ1003に対するレスポンスとして出力(図3、ステップ2060)される。

【0100】以上の手順で、WWWブラウザ1003が送出した仮想URL1111リクエスト、

http://host1:8080/context

/Edit?method=summary&url1
=http%3A%2F%2Fhost0%2Fpage01.html

に対する処理が実行される。

【0101】〔実施例2〕上述した実施例1の構成では、図2に示す構成において1つの処理サーバ、例えば処理サーバ1001、または処理サーバ1002において手続き処理を実行することが可能な構成について説明した。この第2実施例では、複数の処理サーバが協業して各種処理を実行してWWWブラウザの要求に対応してリスポンスを実行する構成を説明する。

【0102】第2の実施例を説明する。第1の実施例で示した図2の構成において、仮想URLを原料のURLとして扱うことにより2つのサーバ（例えば図2の処理サーバ1001、1002）における2つの手続きを用いたファイルの処理が行なえる。2つのコンテキストを用いて、HTML文書からその要約を作成し、それら作成された結果を合成する例をあげて説明する。

【0103】以下の説明において、処理サーバ1001、1002、コンテキスト1411、1421、URLパーザ1611、1621、スナップショットマネージャ1511、1521の動作する手順は実施例1において説明したと同様であるので、それぞれの詳細な記述は省略する。また、URL、仮想URLに対する前記の略記法を用いる。

【0104】図2に示すように、処理サーバ1001、1002にはそれぞれHTML文書中を編集するクラスEditがあり、Editクラスには一つのHTML文書の一部を抽出して一つのHTML文書として出力するメソッドsummaryがあり、複数のHTML文書を合成して一つのHTML文書として出力するメソッドmergeが含まれる。

【0105】原料ファイル1801、1802、1803、1804がWWWサーバ1000上にあり、これらのURLを前記した実施例1中で述べた略記法に従い、「A」、「B」、「C」、「D」とする。サーバ1001上のコンテキスト1411が起動するクラスEditのメソッドsummary1211による処理を「f」と表す。サーバ1002上のコンテキスト1421が起動するクラスEditのメソッドmerge1222による処理を「g」と表す。

【0106】サーバ1002が仮想URL：g（f（A）、f（B）、f（C）、f（D））に対するHTTPリクエストを受信すると、サーバ1002のコンテキスト1421は、サーバ1002のスナップショットマネージャ1521にg（f（A）、f（B）、f（C）、f（D））に対応するスナップショットの有効性を問い合わせる。サーバ1002のスナップショットマネージャ1521は、前記図6、図7の手順にしたがって動作し、必要ならばg（f（A）、f（B）、f（C）、f（D））から、サーバ1001への4つの仮想URL：f（A）、f（B）、f（C）、f（D）を取り出して、それらの仮想URLについて、それぞれスナップショットマネージャ1511に対しスナップショットの有効性を問い合わせるリクエストを送出する。

【0107】スナップショットマネージャ1511は、それぞれのリクエストを受けとり、リクエストごとに独立に有効性の判定をする。前記図6、図7に示した手順にしたがって判定をおこない、それぞれのリクエストごとにスナップショットの有効性の答えを返す。

【0108】サーバ1002のスナップショットマネージャ1521は、図6、図7に示した手順に従い、必要ならば4つの仮想URL：f（A）、f（B）、f（C）、f（D）に対するスナップショットの有効性の答えを受信し、g（f（A）、f（B）、f（C）、f（D））に対応するスナップショットの有効性を計算し、コンテキスト1421に対し答えを返す。

【0109】もし、スナップショットが無効の場合は、コンテキスト1421、1411がそれぞれ図4のステップにしたがって動作し、仮想URL：g（f（A）、f（B）、f（C）、f（D））によって起動される処理の結果のオブジェクトを生成し、スナップショットとして格納する。以上のステップにより、WWWブラウザ1003は、サーバ1000上の4つの原料ページに対して要約を抽出する処理（merge）をした結果を、一つのHTML文書中に合成したHTML文書を受信する。

【0110】また、処理が完了した後、WWWブラウザ1003から同じ仮想URLによるリクエストが送出された場合に、WWWサーバ1000上の原料ページ1801、1802、1803、1804がいずれも更新されていなければ、すでに作成された処理の結果を出力することができる。また、原料ページのうちの一つ以上が更新されていれば、更新されたページを入力とした処理の結果のスナップショットが無効と判定され、処理が再度起動されてスナップショットが再び生成され、その結果が出力される。

【0111】例えば、仮想URL：g（f（A）、f（B）、f（C）、f（D））によるリクエストが送出され、仮想URL：g（f（A）、f（B）、f（C）、f（D））に対応するスナップショット1321が生成され、4つの仮想URL：f（A）、f（B）、f（C）、f（D）に対応するスナップショットとして、それぞれスナップショット1311、1312、1313、1314が生成されたとする。

【0112】その後、URL：Aで表される原料ファイル1801が更新された後に再び仮想URL：g（f（A）、f（B）、f（C）、f（D））によるリクエストが送出された場合、スナップショットマネージャ1511は、スナップショット1311は無効、スナップ

ショット1312, 1313, 1314は有効という答えを返す。また、スナップショットマネージャ1521はスナップショット1321は無効という答えを返す。その結果、コンテキスト1421は仮想URL: g (f (A) , f (B) , f (C) , f (D)) によるメソッドmerge1222の処理をふたたび起動するが、メソッドmerge1222の入力のうち、3つの仮想URL: f (B) , f (C) , f (D) による処理は実行されず、それぞれに対応した3つのスナップショット1312, 1313, 1314が入力される。仮想URL: f (A) による処理は実行され、スナップショット1311が更新され、メソッドmerge1222に入力される。

【0113】このように、利用者が原料ファイルの更新を反映した最新の処理結果を得たい場合、最新の処理結果を得るために必要な最小限度の処理を行ない、それ以外は以前の処理の結果のスナップショットを利用するため、リクエストに対する応答時間を短縮することが可能になる。

【0114】また、信頼性の低いネットワーク上で複数のサーバ上のコンテキストが起動する処理を組み合わせる一つの処理として利用する際、いずれか一個以上の通信が失敗すると、最終的な処理結果が得られない。また、そのような場合、全体のすべての処理を再試行すると、やはり一個以上の通信が失敗する場合には処理結果がえられない。本発明においては、そのような場合に、スナップショットオブジェクトとして一部の処理の結果が保持されるため、再試行により処理結果が得られる確率が増える。

【0115】上述した例では、コンテキストが2つの場合を記述したが、3つ以上の場合でも同様の手順にしたがって処理を組み合わせることが可能である。

【0116】〔実施例3〕本発明の第3実施例として、修飾されたURLの中に含まれた原料ファイル名以外のデータに依存して、スナップショットの有効性を判定する構成例を説明する。システム構成は実施例1と同様、図2に示すものである。

【0117】実施例1に述べた構成と基本的には同様のシステム構成を有するが、スナップショットマネージャ1511, 1521がスナップショットの有効性を判定する際、処理サーバ1001, 1002ごとに特有のルール、あるいは指定された手続きごとに特有のルールに従って特定のオブジェクトを調べ、その結果によりスナップショットの有効性を決定する。

【0118】実施例1に述べた文書の合成をする処理を行なう際のメソッドsummaryにおいて、処理結果のHTMLファイル中に常に処理の行なわれた日付を表示する例を述べる。

【0119】WWWブラウザ1003を利用する利用者がその時点で得られる最新の情報による処理結果を得た

いという要求を持っている場合、処理の行なわれる日時により、日付情報の更新が必要になる。よって日付が変わった場合にはすでに作成され、保持されているスナップショットオブジェクトも更新されるべきであるが、実施例1で説明した手順では、日付情報の参照に関する情報を仮想URL中に記述しない限り、このようなスナップショットオブジェクトが更新されるべきかどうかの判定ができない。

【0120】この実施例における構成は図2に示した構成によって実現される。スナップショットマネージャ1511, 1521の動作は図8、図9に示すとおりであり、ステップ5092からステップ5100が実施例1と異なる。

【0121】仮想URL1111によるリクエストを受けた図2に示すサーバ1001のスナップショットマネージャ1511は、スナップショットの有効性を判定する際に、図8、図9のステップ5092において仮想URL1111をキーとして、ファイル名のリストをチェックリストの記述されたハッシュテーブルから取得する。ステップ5096においてスナップショットマネージャはリスト中のファイルが更新されているかどうか調べる。ステップ5100において、前記のファイル名を調べた結果および仮想URL1111中に記述されたファイルから得た結果をあわせてスナップショットの有効性を判定する。

【0122】リスト中の仮想URL1111に対応するファイル名としてシステム時刻の日付を記述したファイルを用意することで前記のWWWブラウザ1003利用者の要求が達成できる。また、この実施例においてチェックリスト中に記述されるファイルは、例えばCGIを用いてプログラムを起動するWWWファイルアクセスのファイル名であってもよい。

【0123】本実施例の手順を用いれば、手続き作成者が、チェックリストにファイル名のリストを格納することで、スナップショットを更新するかどうかのルールを定めることが可能となる。手続きにより生成される処理結果が、どの時点で無効になるかを手続き作成者が把握している場合には、この方法を使用することでスナップショットの有効性の判定がより正確になる。

【0124】〔実施例4〕本発明の第4実施例として、処理サーバ1001, 1002上に保持されているスナップショットの破棄や無効化を指示することができる構成例について説明する。

【0125】図2で示す実施例1と同様のシステム構成において、クライアントが送出する仮想URLのクエリ記述部分中にスナップショットの破棄を指示するパラメータを記述する。例えば、図2中に示される仮想URL1111は、http://host1:8080/context/Edit?method=summary&url1=http%3A%2F%2Fhost

0%2Fpage01.htmlであるが、これにスナップショットのコントロールの命令の記述として、`control=clear`というパラメータ記述を追加して、`http://host1:8080/context/Edit?method=summary&control=clear&url=http%3A%2F%2Fhost0%2Fpage01.html`とする。

【0126】実施例1で述べたURLパーザ1611、1621にスナップショットの破棄を指示するパラメータを取り出す機能を加えると、URLパーザ1611、1621の動作の流れ図は図10のようにあらわされる。実施例1のURLパーザの動作フローとの差異は、ステップ4050からステップ4052にスナップショットコントロールの命令を取り出す動作が加わった点である。

【0127】スナップショットマネージャ1511、1521の動作フローを図11、図12に示す。本実施例において、実施例1のスナップショットマネージャの動作と異なる点は、ステップ5032からステップ5034のステップである。ステップ5032からステップ5034において、仮想URLから前記スナップショットの破棄を指示するパラメータ`clear`が取り出された場合、スナップショットマネージャは、指定されたURLに対応するスナップショットを無効とする動作を実行する。

【0128】その後、コンテキストは、実施例1と同様に動作して、原料ファイルの取得と処理の起動を行ない、新たに作成された処理結果のオブジェクトをハッシュテーブルに保持する。または、HTTPリクエストヘッダ中に、スナップショットの破棄を指示するパラメータを記述することによっても同様の効果が得られる。コンテキストがJWSから渡された`HttpRequest`オブジェクトからHTTPリクエストのヘッダをとりだし、その中にスナップショットの破棄を指示するパラメータが記述されていた場合は、スナップショットマネージャに対し、指定されたURLに対応するスナップショットの破棄を指示することによって行う。

【0129】実施例1～3においては、スナップショットの無効化は、スナップショットを保持しているサーバのスナップショットマネージャの判定によってなされたが、この実施例においては、他のコンピュータからの指示により処理の結果を無効化することができる。そのため、すでに存在するスナップショットではなく処理を再び行なった結果を得たいという利用者の要求を満たすことができ、新しく生成された処理結果をスナップショットとして保持することができる。

【0130】〔実施例5〕さらに本発明の第5実施例として、処理結果のスナップショットオブジェクトを、そ

れを生成したリクエスト固有の識別子(ID)と組にして保持することで、特定のリクエストによって生成されたオブジェクトを指定して取得する構成を説明する。本実施例の構成を図13に示す。図13において処理サーバ1001、1002中のハッシュテーブル中のスナップショット1311、1312...、1321...には識別子(ID)が対応して保持されている。

【0131】先に説明した実施例1においては、同じ原料に対して同じ手続きを適用するリクエストは同一とみなしていた。すなわち、あるリクエストにより処理を行い、結果をスナップショットオブジェクトとして保持したのち、同一のリクエストを再び受信し処理を再び行った場合は、前記後のリクエストによる処理結果は、前記最初のリクエストにより生成されたスナップショットオブジェクトを置き換えて保持される。

【0132】この実施例においては、各リクエストごとに固有の識別子(ID)が割り当てられ、識別子と対応付けてスナップショットオブジェクトが生成され、保持される。クライアントはリクエストに固有の識別子を指定してそれに対応するスナップショットオブジェクトを取得することができ、また、実施例4で述べた手順を用いれば、指定した識別子に対応するスナップショットオブジェクトを破棄し、新たな処理の結果を同じ識別子に対応付けて保持することができる。

【0133】この実施例におけるコンテキストの動作を図14、図15に示す。HTTPリクエストを受けとったコンテキストは、図16に従い動作するURLパーザを用いてURLを解析してオブジェクトを生成するが、URLに識別子(ID)が含まれていないときはリクエストごとに固有のIDを生成して仮想URLに付加する。IDは、一例としてリクエストの発生順序の整数値として実現できる。また、別な一例としてシステム時刻の数値として実現できる。仮想URLの中に、スナップショットのIDを指示するパラメータを付加する。

【0134】例えば、実施例1で述べたクラス`Edit`のHTML文書中の一部を抽出するメソッド`summary`による処理を起動するURL1111は`http://host1:8080/context/Edit?method=summary&url=http%3A%2F%2Fhost0%2Fpage01.html`であるが、これに、IDを付加する記述として`id=`の記述に続けて数字を記述する形式を用いて`http://host1:8080/context/Edit?method=summary&id=1&url=http%3A%2F%2Fhost0%2Fpage01.html`を生成する。

【0135】図14、図15にコンテキストの動作フローを示す。動作手順について例を挙げて説明する。図13に示す処理サーバ1001のコンテキスト1411が仮想URL1111によるリクエストを受けた場合、ス

ステップ3010においてサーブレットcontextのメソッドserviceが引数のオブジェクトHttpServletRequest, HttpServletResponseを受けとって起動する。

【0136】ステップ3020においてオブジェクトHttpServletRequestのgetRequestedURIメソッドにより、仮想URLに含まれる文字列 /context/Edit?method=merge&arg1=x&arg2=y を取り出す。この文字列を先頭部と合成して仮想URLを再構成する。

【0137】ステップ3030において図17、図18に示した手順で動作するURLパーザ1611を用いて仮想URL1111を解析してVURL1111 (obj) を生成する。ステップ3032においてVURL1111がIDの情報を持っているかどうか調べる。仮想URL1111にIDがついていない場合は、ステップ3034に進む。

【0138】ステップ3034において、ブラウザからのリクエストに対する一意な識別子であるID1を生成し、VURL1111 (obj) に付加する。ステップ3050において、仮想URL1111より取り出した原料のURLが仮想URLかどうかをしらべる。仮想URLであると判定されればステップ3052において、もとの仮想URL1111と対応づけられたID1を原料のURLに付加する。仮想URL1111から取り出される原料のURLは仮想URLではないのでステップ3054に進む。ステップ3054において、VURL1111 (obj) のインスタンス変数に格納されている原料のURLに対してHTTPリクエストを送出し、原料のファイルを取得する。ステップ3060においてすべての原料ファイルを取得したかどうか調べ、すべて取得していなければステップ3050に戻る。

【0139】ステップ3070において取得したファイルを引数として、VURLオブジェクトのインスタンス変数に格納されているクラス名とメソッド名で定まるメソッドを起動しその返り値を処理結果としてオブジェクトを生成する。ステップ3080において処理結果のオブジェクトを仮想URLをキーにして前記ハッシュテーブルに格納する。ステップ3090において処理結果のオブジェクトをServletOutputStreamに書き込む。

【0140】クライアントが別のリクエストを送出する際にIDを付加した仮想URLを用いると、コンテキストは、図14のステップ3032において、IDが付加されているために、3040に進む。ステップ3040において、コンテキストはスナップショットマネージャにVURLオブジェクトを渡し、仮想URL及びIDに対応するスナップショットが有効かどうか問い合わせる。

【0141】有効なスナップショットがあった場合、ステップ3100において、そのスナップショットオブジェクトをハッシュテーブルから取り出す。ステップ3110においてオブジェクトをServletOutputStreamに書き込む。

【0142】URLパーザは、図16に示す動作フローに従って動作する。実施例1で説明したURLパーザの動作フロー（図5参照）との差異は、図16のステップ4050と、ステップ4052が加わった点と、ステップ4090の出力形態にIDが加わった点である。

【0143】ステップ4010～ステップ4040までは実施例1と同様の動作であるので説明を省略する。ステップ4050において、取り出された<propertyName>=<propertyValue>の形式の文字列について、propertyNameがidであるかどうかの判定をおこなう。判定がYesであった場合は、ステップ4052においてpropertyValueの値であるIDを取り出す。

【0144】その後のステップ4054～ステップ4080は実施例1と同様の動作である。ステップ4090では、仮想URLを表すVURLクラスのオブジェクトを生成し、クラス名Edit、メソッド名summary、ID、原料のURLをインスタンス変数の値としてセットする。

【0145】スナップショットマネージャは、図17、図18にしたがって動作する。実施例1のスナップショットマネージャの動作と異なる点は、ステップ5020において、仮想URLおよびIDが一致するスナップショットオブジェクトが存在するかどうかでスナップショットの有効性を判定する点である。

【0146】実施例の細部に関しては、記述した以外の方法を用いても実現可能である。サーバからクライアントにID情報を伝える方法は、HTMLページ中に記述する方法に限らない。IDを付加した仮想URLの情報をHTTPレスポンスのヘッダに含めてクライアントに渡してもよい。

【0147】また、IDの生成方法は、時刻とサーバのホスト名によって生成するものであってもよい。クライアントがリクエストを送出する際にIDを付加した仮想URLを用いることで、特定のリクエストで生成されたオブジェクトを取得することができる。

【0148】この実施例で述べた手順により、特定の時点での処理の結果をサーバ上に保持しておき、指定して再度取得することが可能になる。

【0149】〔実施例6〕次に、第6実施例として、サーバ上に保持された複数のスナップショットオブジェクトのうち、生成した時刻の範囲を指定して取得する構成例について説明する。

【0150】コンテキストの動作は実施例1と同様の手順であるが、ただし、処理の結果が生成されたときにそ

の時刻を記録し、スナップショットオブジェクトとともに保持する点異なる。

【0151】スナップショットマネージャの動作の流れ図は図19、図20に示す。スナップショットマネージャは、図19、図20のステップにおいて、仮想URL中にスナップショットオブジェクト生成時刻の範囲が指定されるかどうか調べ、指定されていたらステップ5034においてスナップショットオブジェクトの生成時刻を取得し、ステップ5036において指定された時刻の範囲にあるかどうかにより判定する。範囲外の場合はステップ5038においてスナップショット無効の結果を返す。これらのステップが加わる点が実施例1のスナップショットマネージャの動作と異なる。時刻の範囲が指定されていなければ、実施例1と同様に動作する。

【0152】時刻の範囲の指定の例としては、例えばUTC (Coordinated Universal Time) 時刻を1970年からミリ秒単位で数えた数値で時刻を表現して、例えば895809110000から895809120000までという時刻の範囲を、仮想URLのクエリ文字列中に `f t i m e = 8 9 5 8 0 9 1 1 0 0 0 0 & t t i m e = 8 9 5 8 0 9 1 2 0 0 0 0` と記述する。URLパーザは前記の動作と同様にしてこの文字列を取り扱い指定時刻を認識、解釈する。

【0153】このようにして、生成時刻の正確な指定でなく、ある程度の幅を持った指定をして、スナップショットを利用するか、あるいは処理を起動して新しく結果を生成するかの指定をすることが可能になる。

【0154】また、生成時刻に限らず、スナップショットされたオブジェクトの性質を表し、数値表現可能なパラメータをスナップショットオブジェクトと関連付けて保持することにより、同様の手順で、値がある範囲内ならスナップショットを利用する、という条件の指定が行なえる。

【0155】【実施例7】次に、第7実施例としてコンテキストによって起動される処理結果が非同期的に生成される構成例を説明する。実施例5と同様の手順で動作するシステムを用いる。ただし、仮想URL中には手続きの呼び出し形式を指定でき、呼び出しが非同期的なものであることを指定できる。指定方法の例としては、URLのクエリ中に、`& i n v o k e = a s y n c` と指定して、

`http://host1:8080/context/`
`Edit?method=summary&i`
`nvoke=async&url=http%3A%2F%2F`
`host0%2Fpage01.htm`

のようなURLを用いる。

【0156】コンテキストは図21、図22にしたがって動作する。ステップ3060において、仮想URLに呼び出しが非同期である指定があるかどうか調べる。非

同期の指示があった場合、コンテキストは、ステップ3062において原料の取得をおこなって手続きを起動し、別なスレッドで動作させる。次にステップ3064においてクライアントに出力する結果オブジェクトに対応するIDを含んだ情報を生成する。IDをクライアントに対し出力するには、HTMLページ中にハイパーリンクとして記述するなどの方法が利用できる。次にステップ3066において生成した情報を出力し、HTTPのアクセスが終了してコネクションが切断される。

【0157】前記別スレッドの処理が終了後、ステップ3068において、結果オブジェクトを、仮想URLとIDとの組にして保持する。以上のステップ3060から3068以外は実施例5のコンテキストと同様に動作する。

【0158】WWWクライアント利用者、または他のコンテキストはIDを受けとってHTTPのアクセスを終了し、他の処理をおこなうことができる。その後、実施例5で述べたごとく、仮想URLにIDを付加してHTTPリクエストを行なうと、その時点で処理サーバにおけるスナップショット生成処理が終了していれば処理結果を得ることができる。

【0159】先に説明した他の実施例においては、呼び出した処理が終了して結果が生成されるまでWWWクライアントとのコネクションが維持され、クライアントはその間、出力データを待機することが要求され、他の作業をできなかったが、この例においては、処理終了までに時間がかかる場合に処理を呼び出した側で並列に別な処理を行うことが可能になる。

【0160】処理が実行され生成されたスナップショットはIDと対応づけられて処理サーバにおいて保持されており、クライアントはIDの付加された新たなHTTPリクエストによって先に処理を要求したデータをIDによる識別により獲得することが可能となる。

【0161】【実施例8】次に、本発明の第8実施例として、クライアントがリクエスト時に指定し、仮想URL中に記述された条件を、処理サーバのコンテキストが変更して、クライアントの要求データの代替となりうるスナップショットを探し、そのスナップショットを出力する構成例について説明する。

【0162】この実施例構成では、実施例5と同様の手順で動作するシステムを用いる。ただし、指定したIDおよび仮想URLに対応するスナップショットが存在しないときには、IDが異なるが同一の仮想URLに対応するスナップショットを探して有効なスナップショットが存在したときはそれをクライアントに対して出力する構成を持つ。

【0163】コンテキストは実施例5と同様に図14、図15にしたがって動作する。ステップ3040において、スナップショットマネージャに対し仮想URLとIDに対応するスナップショットの有効性を問い合わせ

る。

【0164】スナップショットマネージャの動作は図23、図24に示す通りであり、以下に述べる点以外は実施例5のスナップショットマネージャと同じ動作をする。ステップ5020において仮想URLおよびIDに対応するスナップショットオブジェクトがなかったとき、ステップ5022に進み、仮想URLのみが一致するスナップショットがあるかどうかを調べる。もしなかったときは5030にすすみ、スナップショット無効の結果を論理値の偽として返す。スナップショットがあったときは、ステップ5040に進み、実施例5と同様にスナップショットの有効性を判定するが、この場合には対象となるスナップショットはステップ5022において調べた仮想URLのみが一致し、IDについての一致は保証されていないスナップショットである。

【0165】図23、図24のステップ5022におけるスナップショットマネージャの動作は、ここで記述したようなIDの一致しないスナップショットを代替として探す動作に限らず、たとえば実施例5において記述した範囲の指定があるような仮想URLについて、範囲の指定をなくす、あるいは広げるような動作をさせることも可能である。

【0166】前記の実施例5においては、仮想URLおよびIDが一致しないとスナップショットを利用できなかったが、この実施例の手順によって、利用者の、なるべくIDの一致するスナップショットオブジェクトを利用したいが、もし無い場合には代替のオブジェクトを得たいという要求を満たすことが可能になる。

【0167】

【発明の効果】上述のように本発明の分散ファイル処理装置および分散ファイル処理方法によれば、原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名等で修飾して構成した仮想URLに基づく処理を行うコンテキスト手段と、コンテキスト手段において処理を行った原料ファイルの処理結果を保持する結果保持手段と、結果保持手段の保持する処理結果の有効性を判定する結果管理手段とを有することにより、結果保持手段の有する処理結果の有効性判定を様々な態様で実行することが可能になり、要求に対して迅速に有効な処理結果を抽出し出力することが可能となる。

【0168】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、原料ファイル名および手続き名等の識別データ、さらに手続き処理の処理実行日時を示す処理日時識別データ、手続き処理の実行を要求した要求固有の処理要求識別データ等に基づいて処理結果の有効性を判定することができ、処理結果に対する信頼度の高い有効性判定が実現される。

【0169】さらに、本発明の分散ファイル処理装置お

よび分散ファイル処理方法によれば、複数のコンテキスト手段、結果保持手段、管理手段をネットワークに接続した構成において、処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の他の処理結果保持手段を管理する管理手段に処理結果の有効性についての問い合わせを実行することが可能であり、ネットワーク上の資源の有効な利用が達成される。

【0170】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行することが可能であり、処理結果に対応する処理結果識別子を生成して結果保持手段に保持することにより、処理結果の取り出し、出力を正確に実行することが可能となる。

【0171】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、要求された修飾ファイル名の指定と完全一致しないが、部分一致する処理結果が結果保持手段に含まれる場合、部分一致する処理結果を出力する構成としたことにより、クライアント要求により迅速に応答することの可能なシステムが提供される。

【図面の簡単な説明】

【図1】 本発明の分散ファイル処理装置を適用したネットワーク構成を示す図である。

【図2】 本発明の分散ファイル処理装置における実施例1の詳細構成を示す図である。

【図3】 本発明の分散ファイル処理装置、実施例1における処理サーバ全体の動作フローを示す図である。

【図4】 本発明の分散ファイル処理装置、実施例1におけるコンテキストの動作フローを示す図である。

【図5】 本発明の分散ファイル処理装置、実施例1におけるURLパーザの動作フローを示す図である。

【図6】 本発明の分散ファイル処理装置、実施例1におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図7】 本発明の分散ファイル処理装置、実施例1におけるスナップショットマネージャの動作フロー（その2）を示す図である。

【図8】 本発明の分散ファイル処理装置、実施例3におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図9】 本発明の分散ファイル処理装置、実施例3におけるスナップショットマネージャの動作フロー（その2）を示す図である。

【図10】 本発明の分散ファイル処理装置、実施例4におけるURLパーザの動作フローを示す図である。

【図11】 本発明の分散ファイル処理装置、実施例4におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図12】 本発明の分散ファイル処理装置、実施例4におけるスナップショットマネージャの動作フロー（その2）を示す図である。

【図13】 本発明の分散ファイル処理装置における実施例5の詳細構成を示す図である。

【図14】 本発明の分散ファイル処理装置、実施例5におけるコンテキストの動作フロー（その1）を示す図である。

【図15】 本発明の分散ファイル処理装置、実施例5におけるコンテキストの動作フロー（その2）を示す図である。

【図16】 本発明の分散ファイル処理装置、実施例5におけるURLパーザの動作フローを示す図である。

【図17】 本発明の分散ファイル処理装置、実施例5におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図18】 本発明の分散ファイル処理装置、実施例5におけるスナップショットマネージャの動作フロー（その2）を示す図である。

【図19】 本発明の分散ファイル処理装置、実施例6におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図20】 本発明の分散ファイル処理装置、実施例6におけるスナップショットマネージャの動作フロー（その2）を示す図である。

【図21】 本発明の分散ファイル処理装置、実施例7におけるコンテキストの動作フロー（その1）を示す図である。

【図22】 本発明の分散ファイル処理装置、実施例7におけるコンテキストの動作フロー（その2）を示す図である。

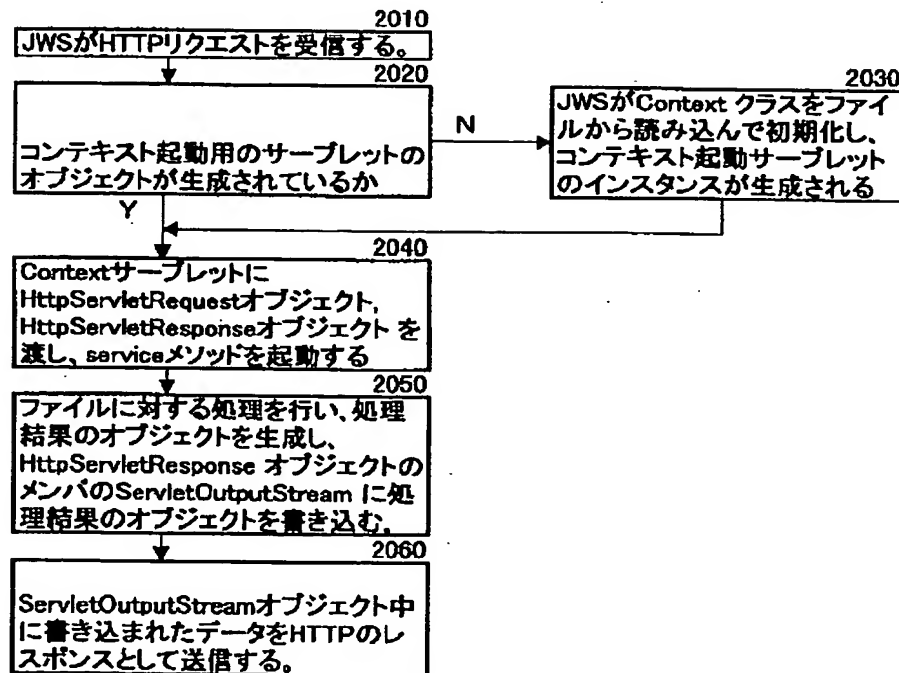
【図23】 本発明の分散ファイル処理装置、実施例8におけるスナップショットマネージャの動作フロー（その1）を示す図である。

【図24】 本発明の分散ファイル処理装置、実施例8におけるスナップショットマネージャの動作フロー（その2）を示す図である。

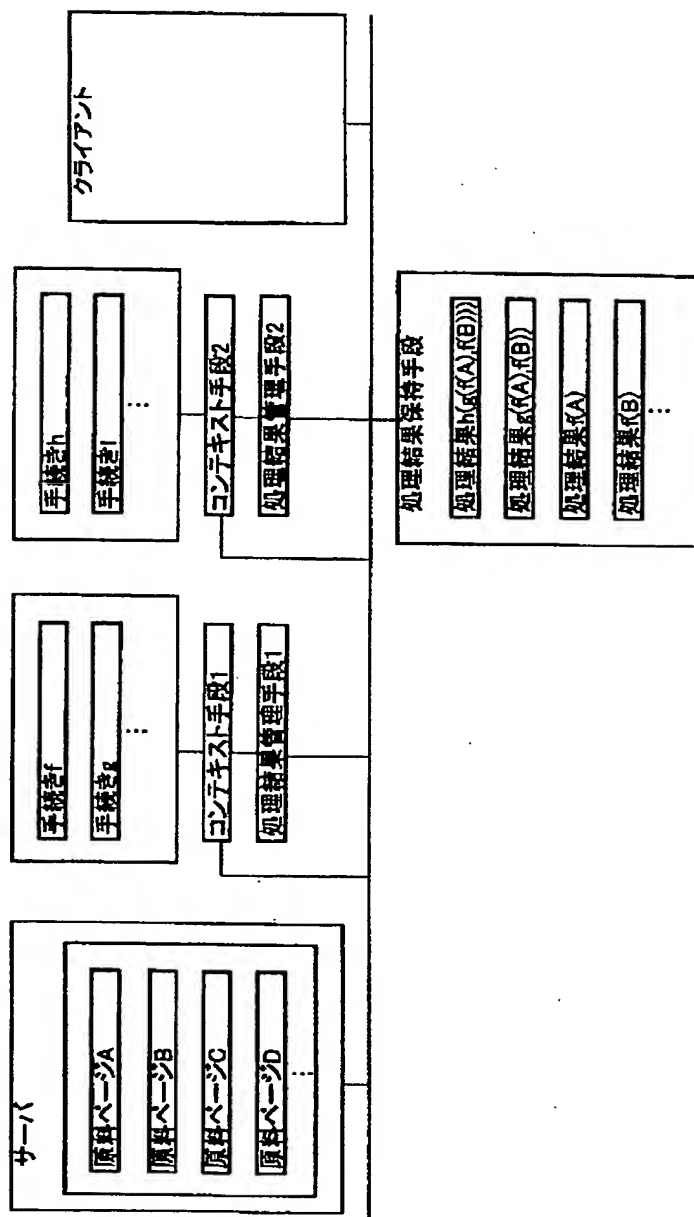
【符号の説明】

1000 WWWサーバ
1001, 1002 処理サーバ
1003 WWWブラウザ
1411, 1421 コンテキスト
1511, 1521 スナップショットマネージャ
1611, 1621 URLパーザ

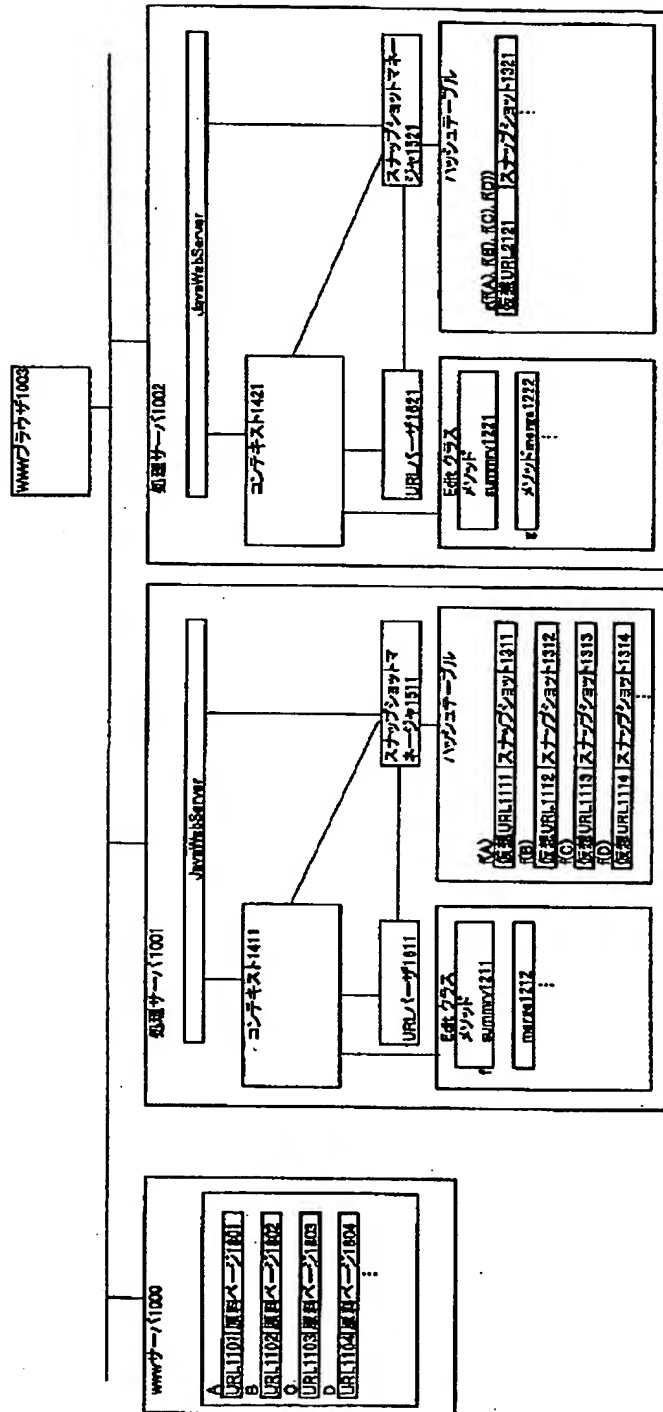
【図3】



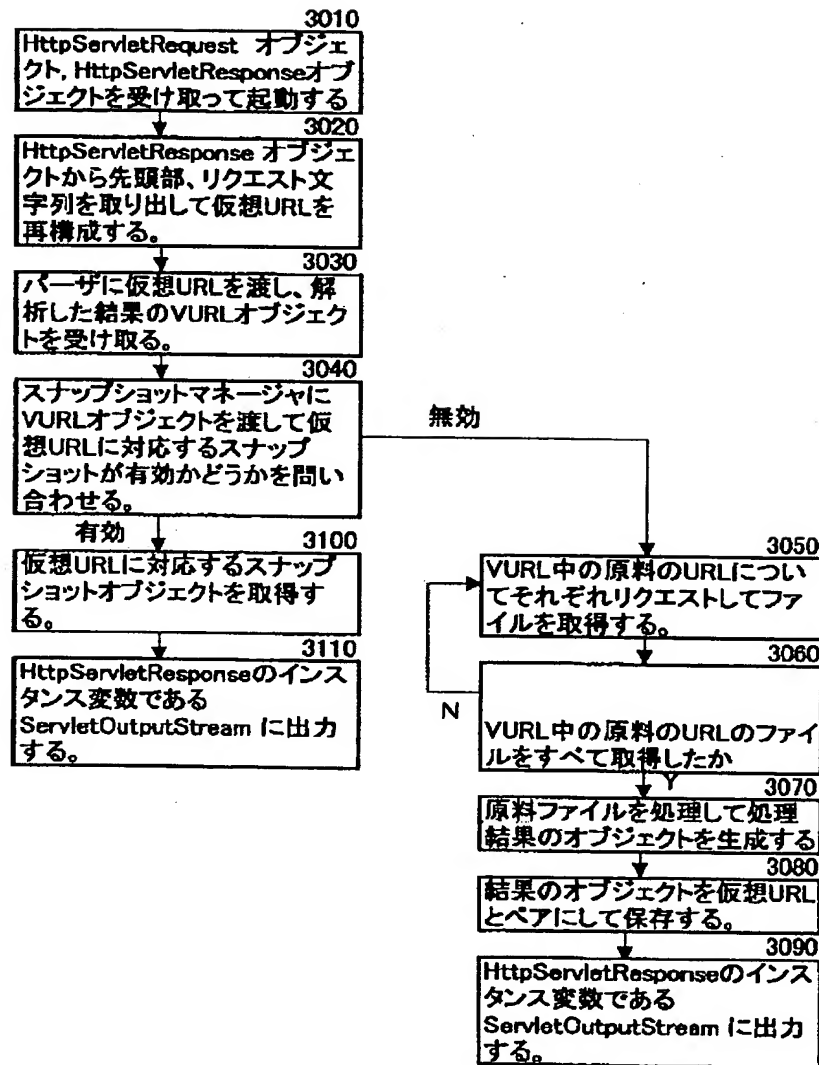
【図1】



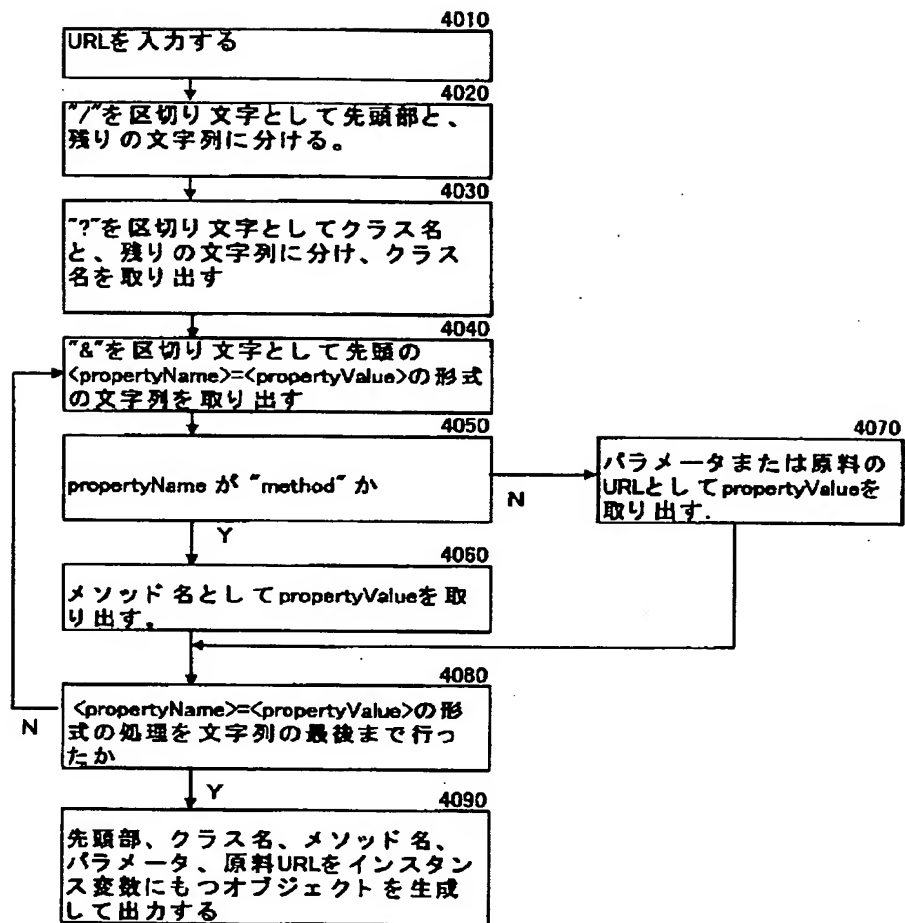
【図2】



【図 4】

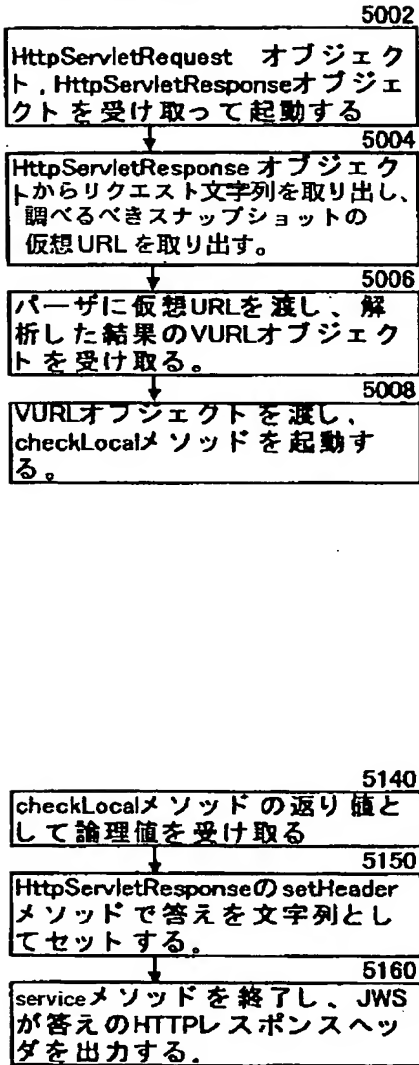


【図5】

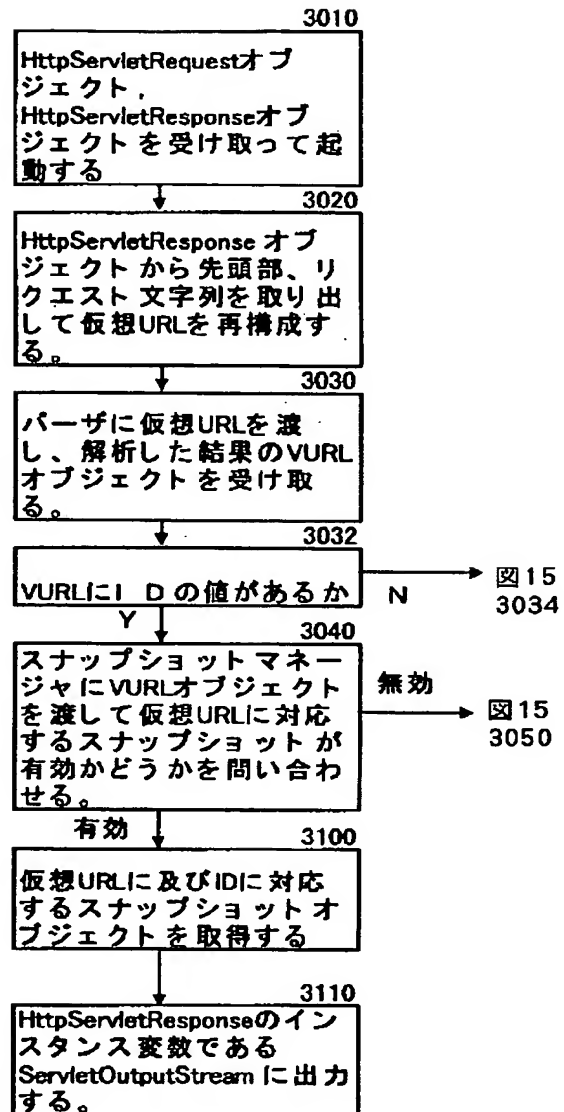


【図6】

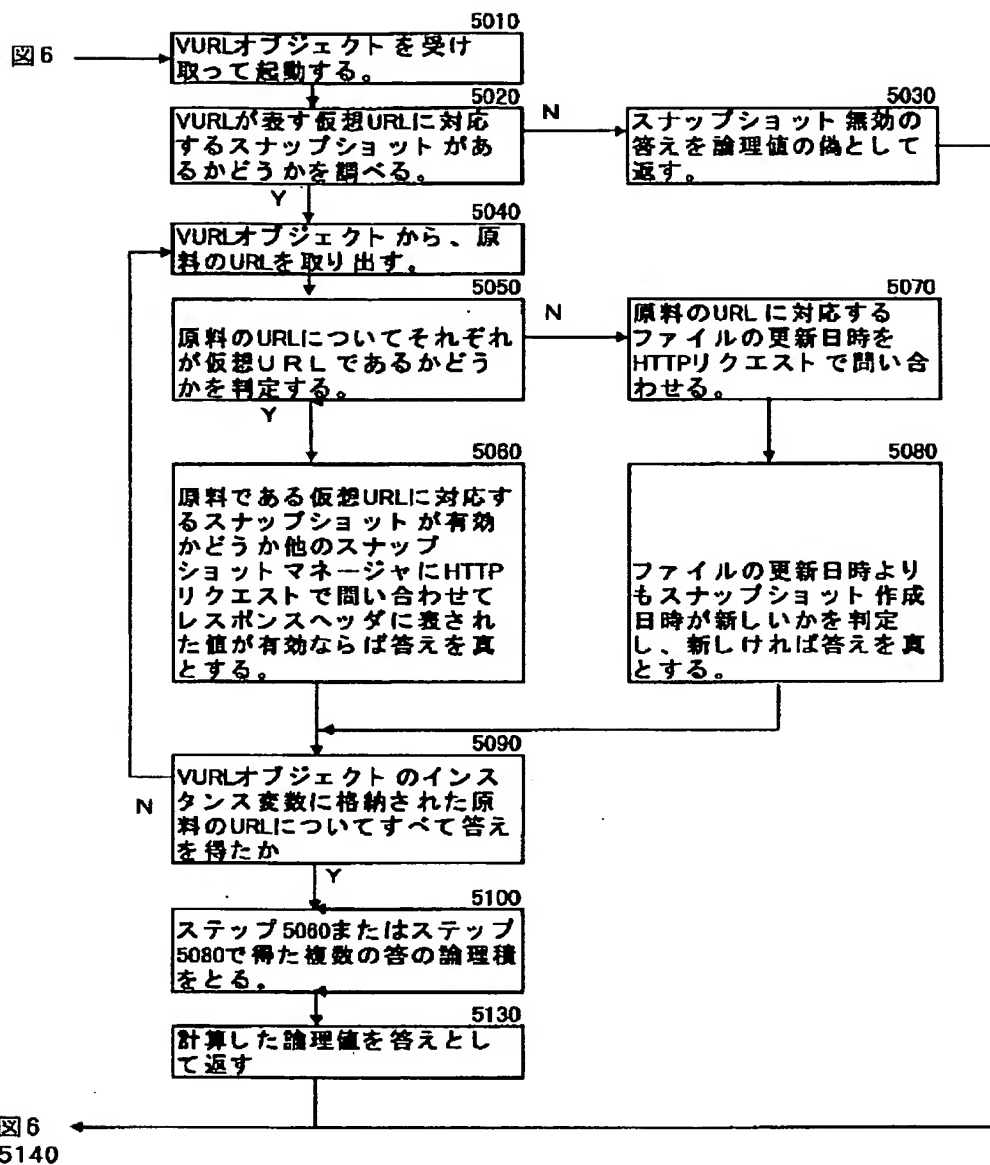
メソッド service



【図14】



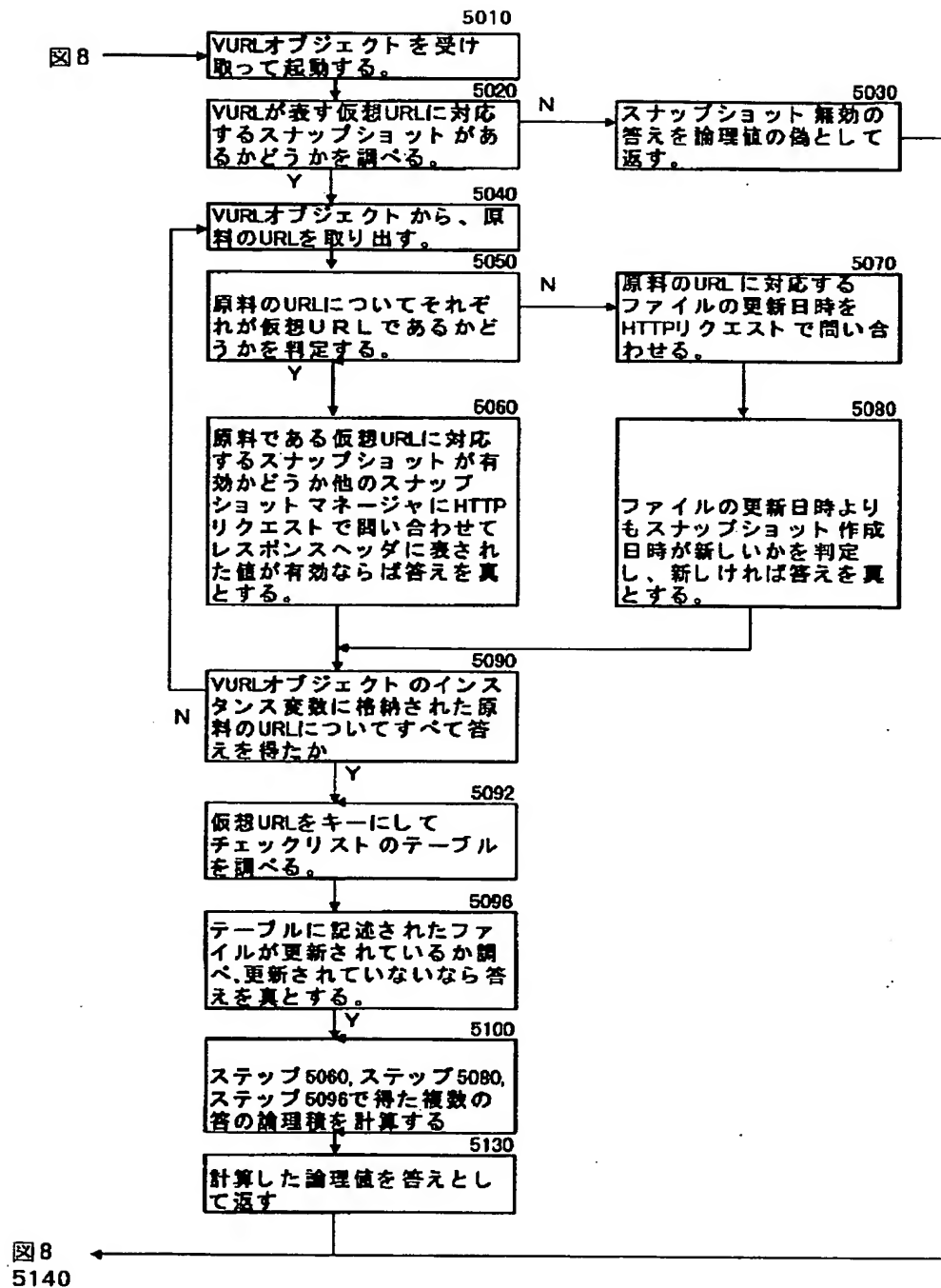
【図7】



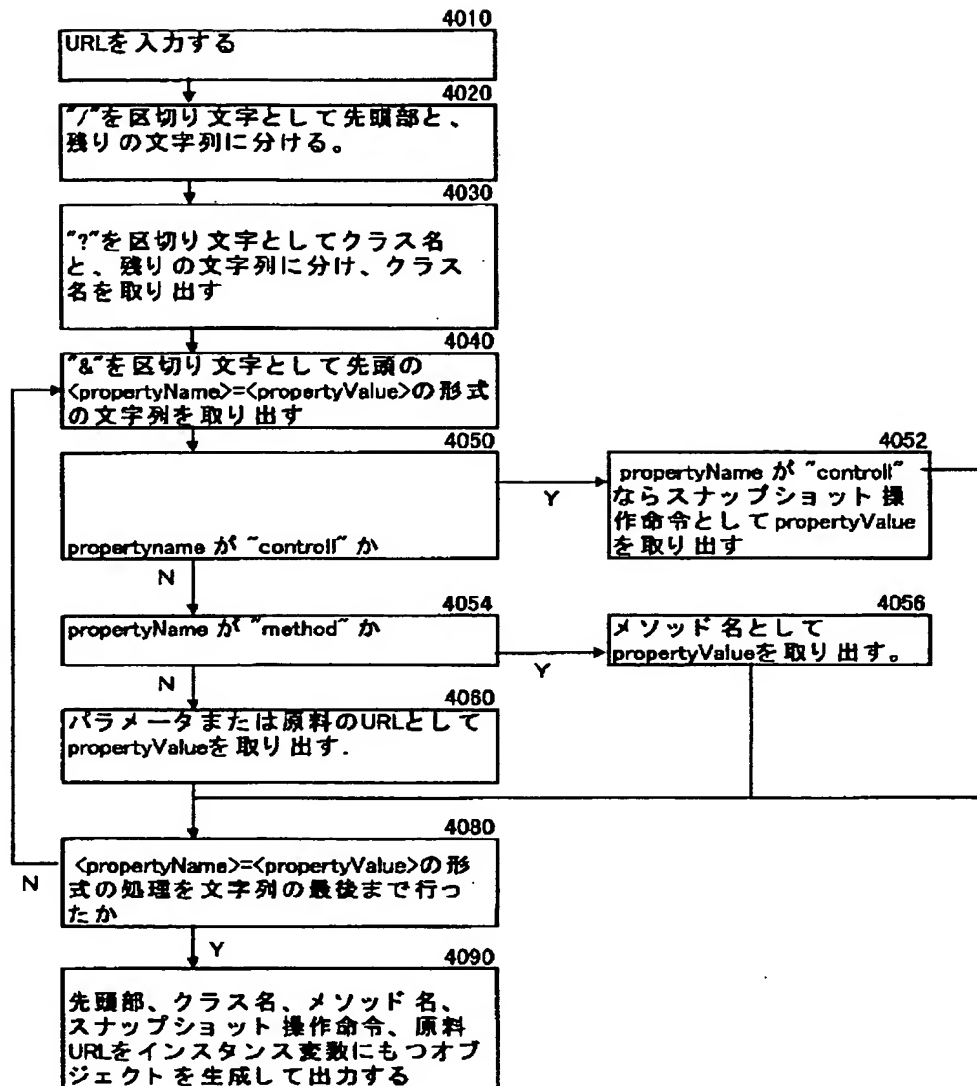
メソッド service



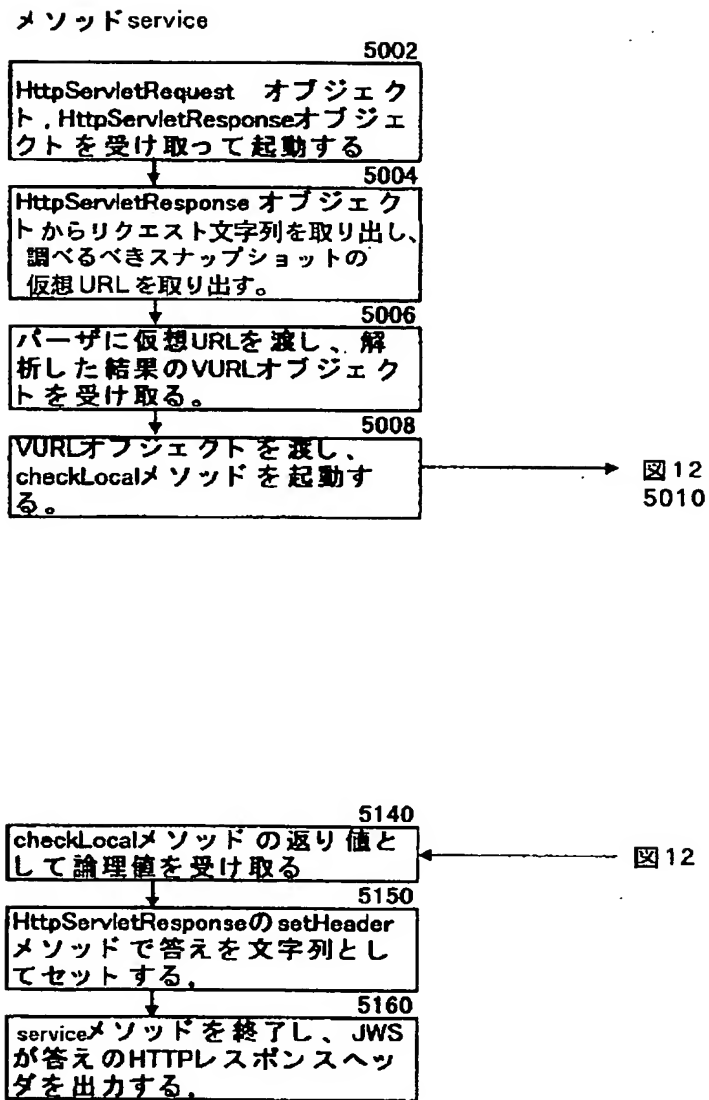
【図9】



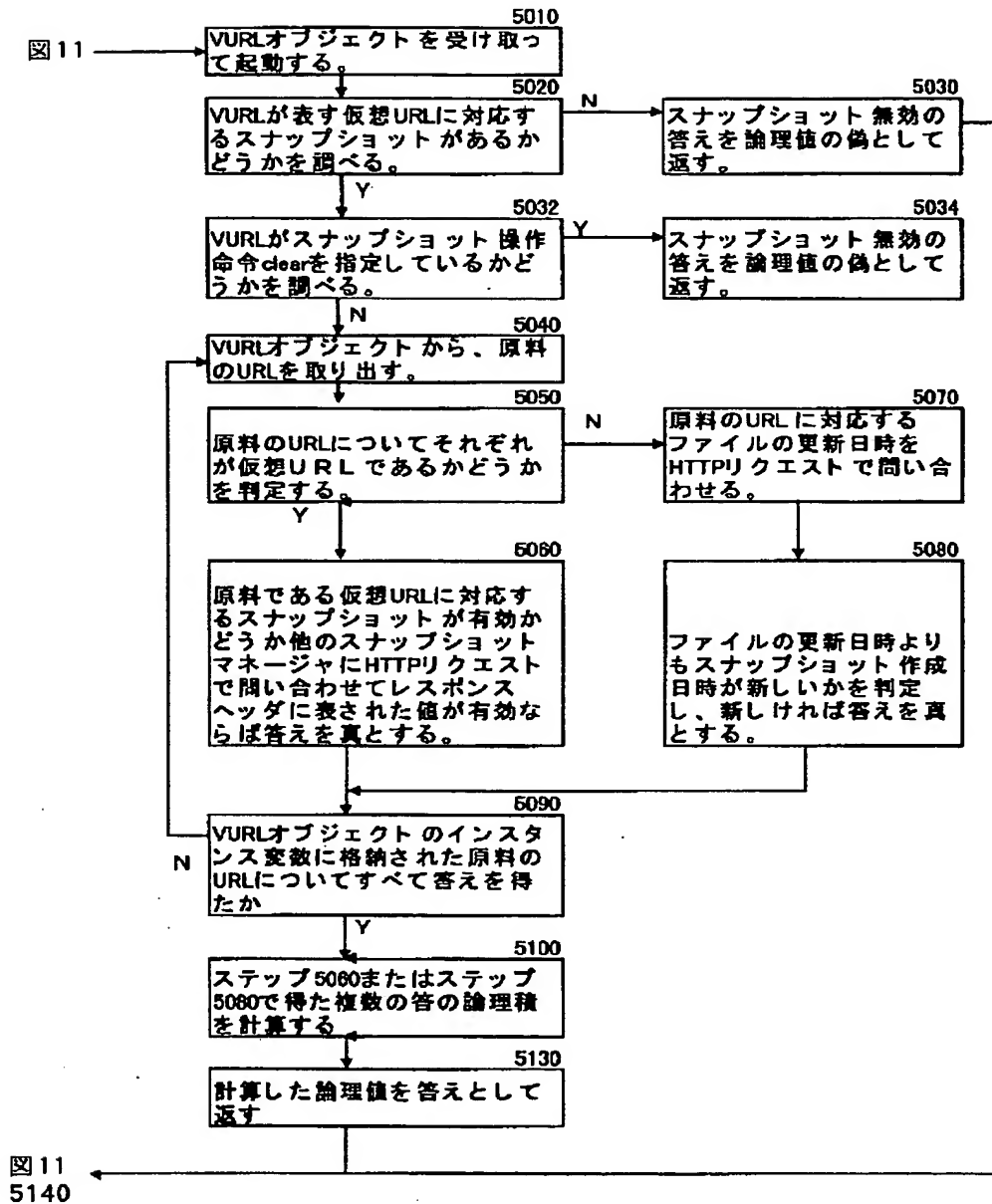
【図 10】



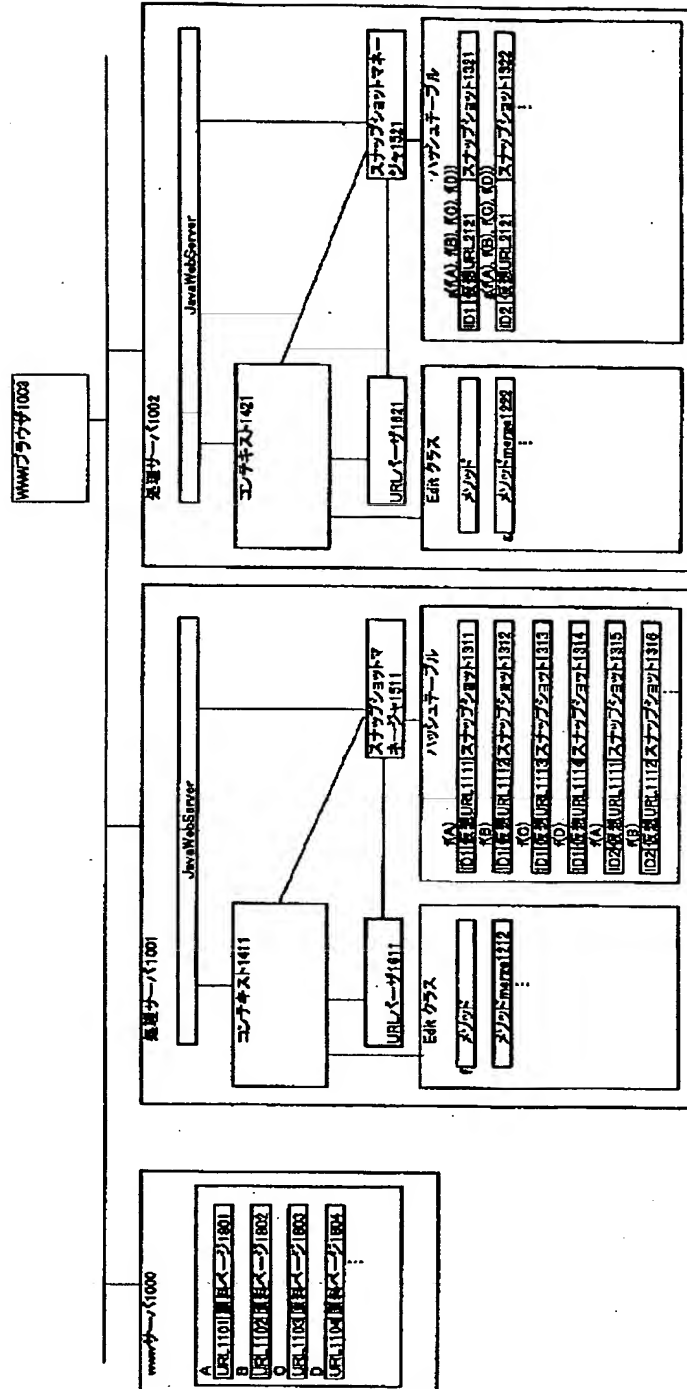
【図11】



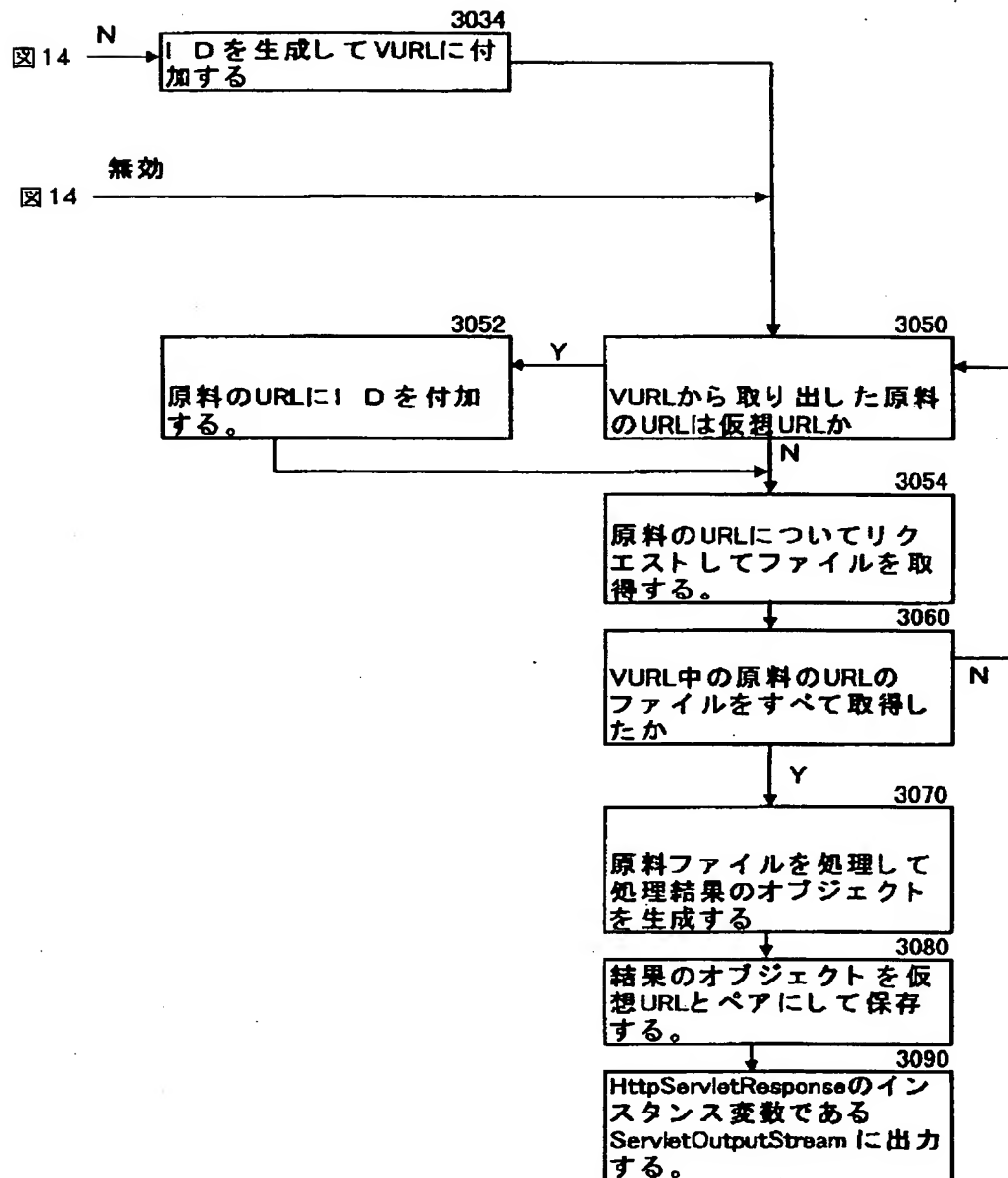
【図12】



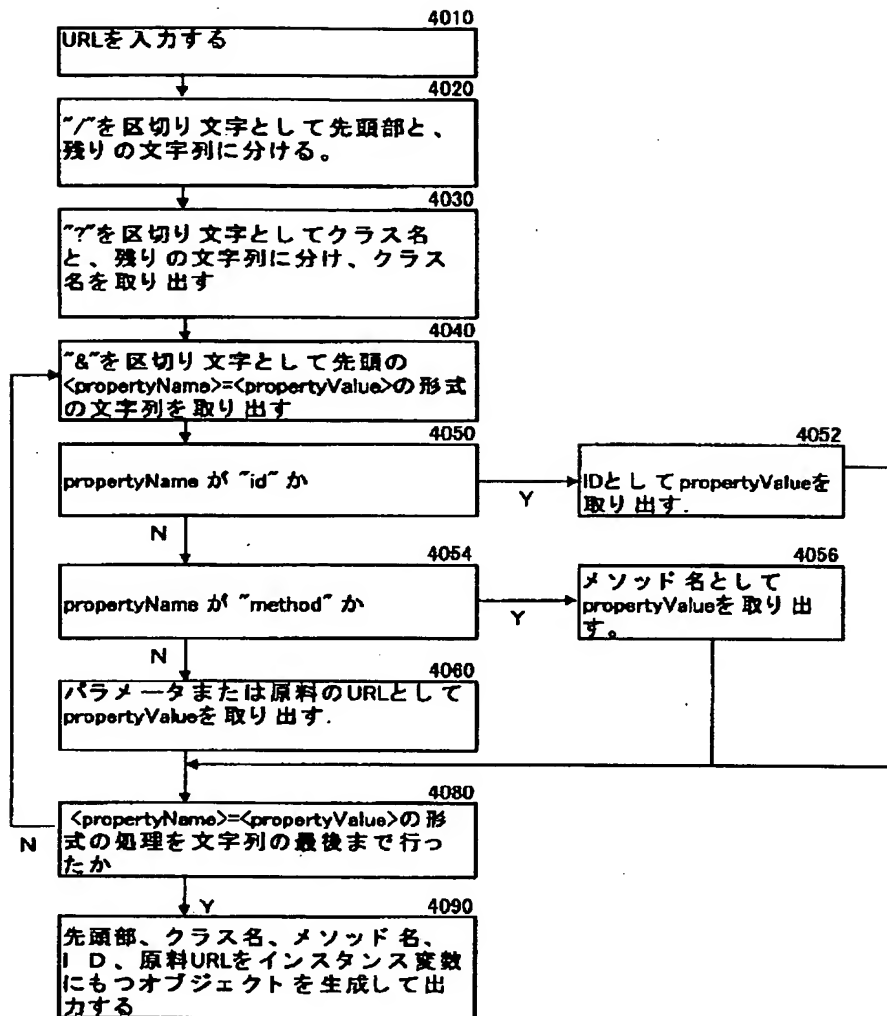
【図13】



【図15】

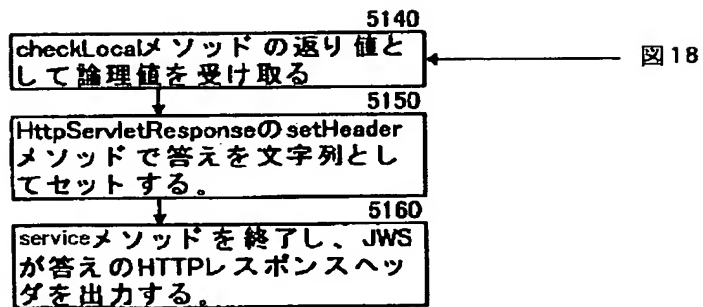
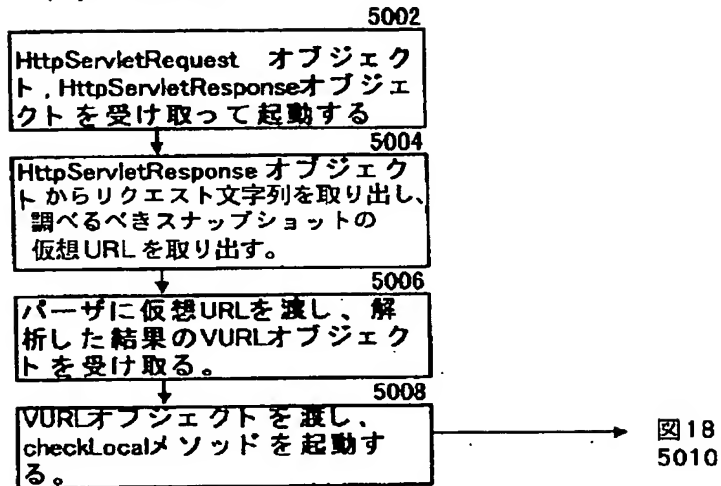


【図16】

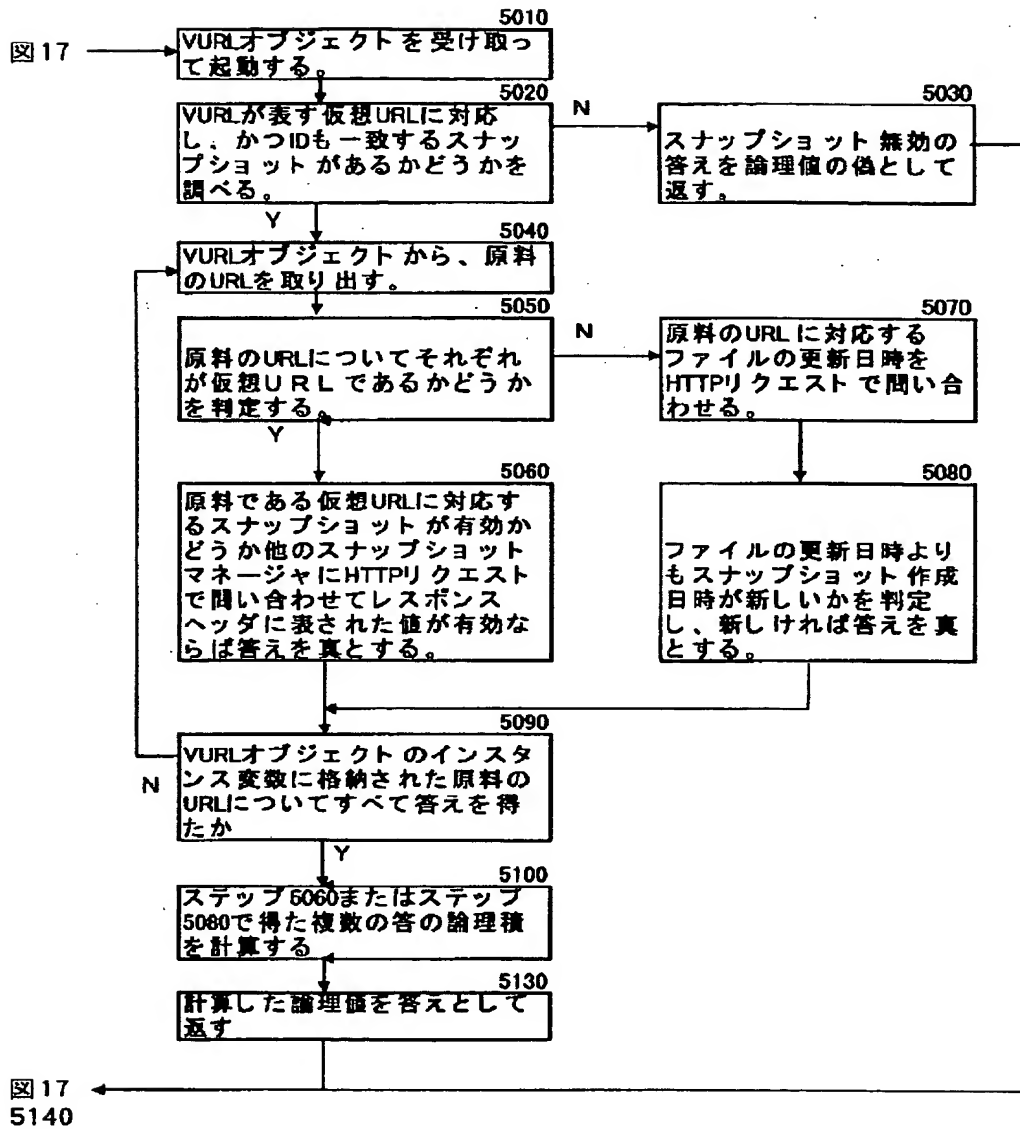


【図17】

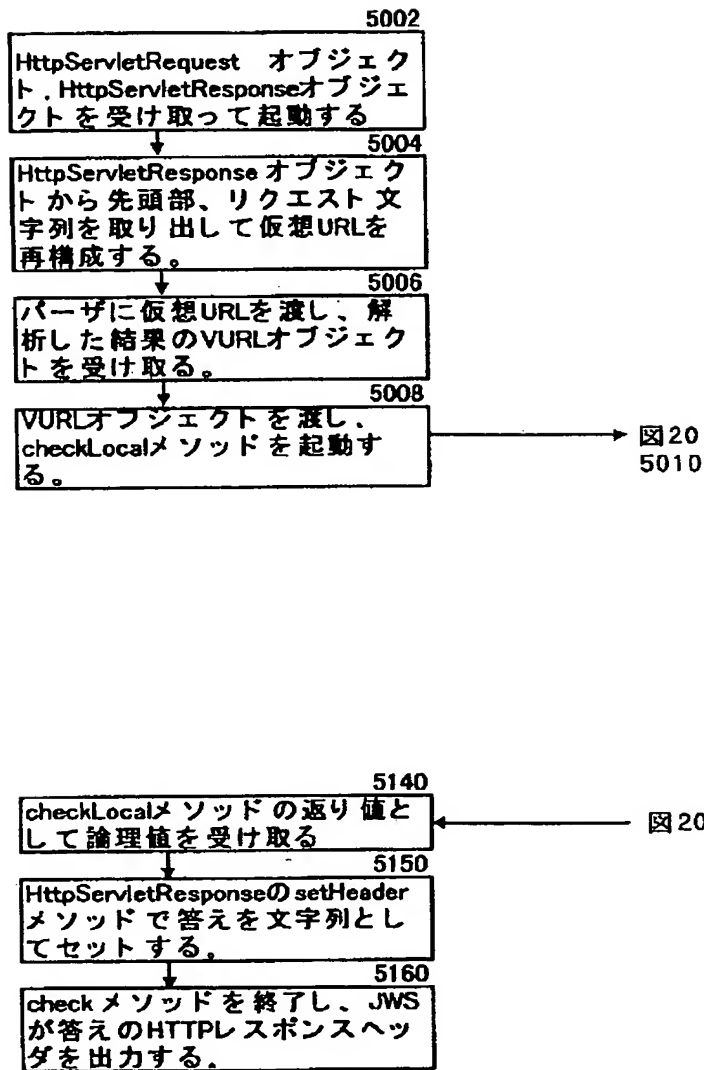
メソッド service



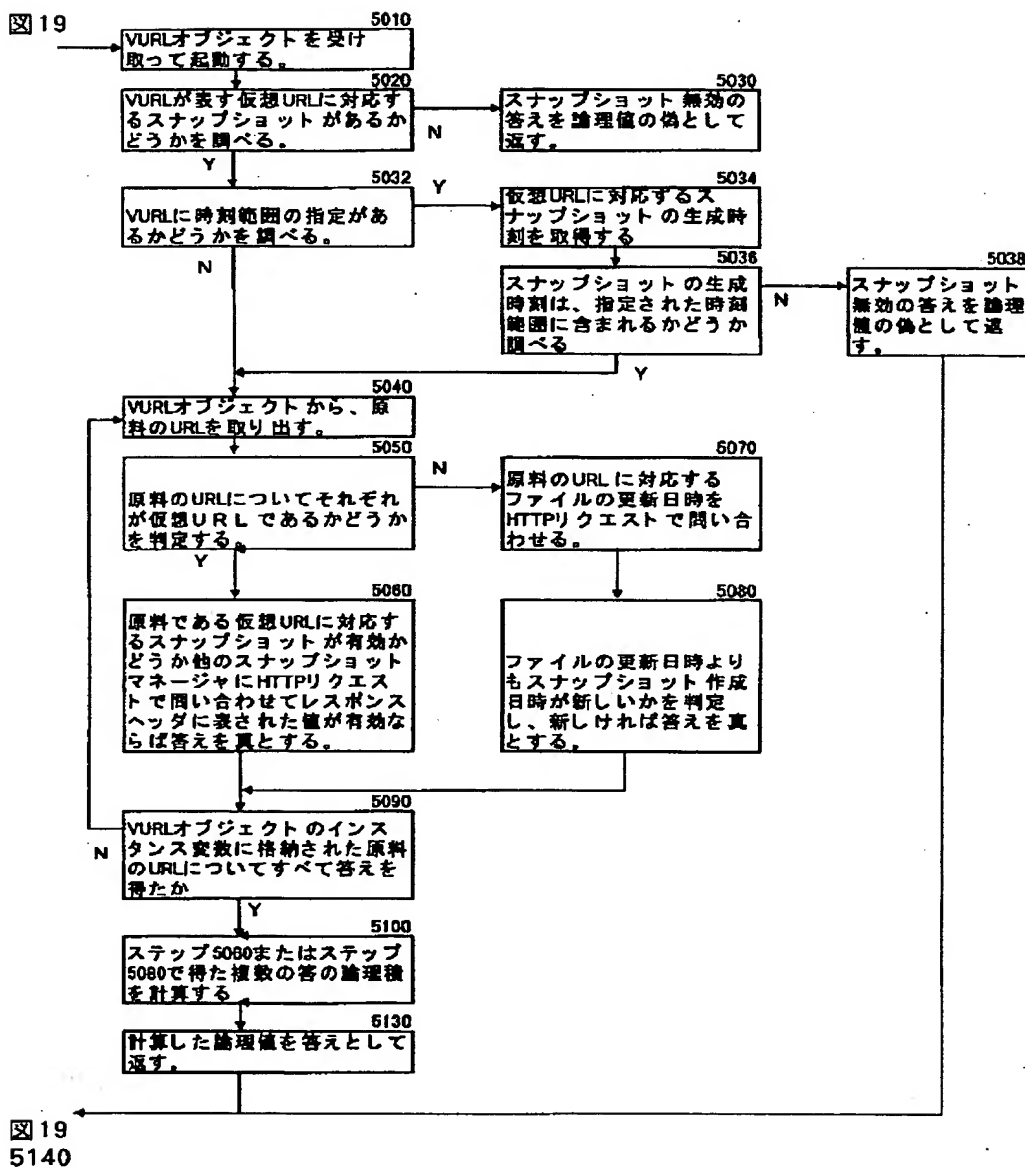
【図18】



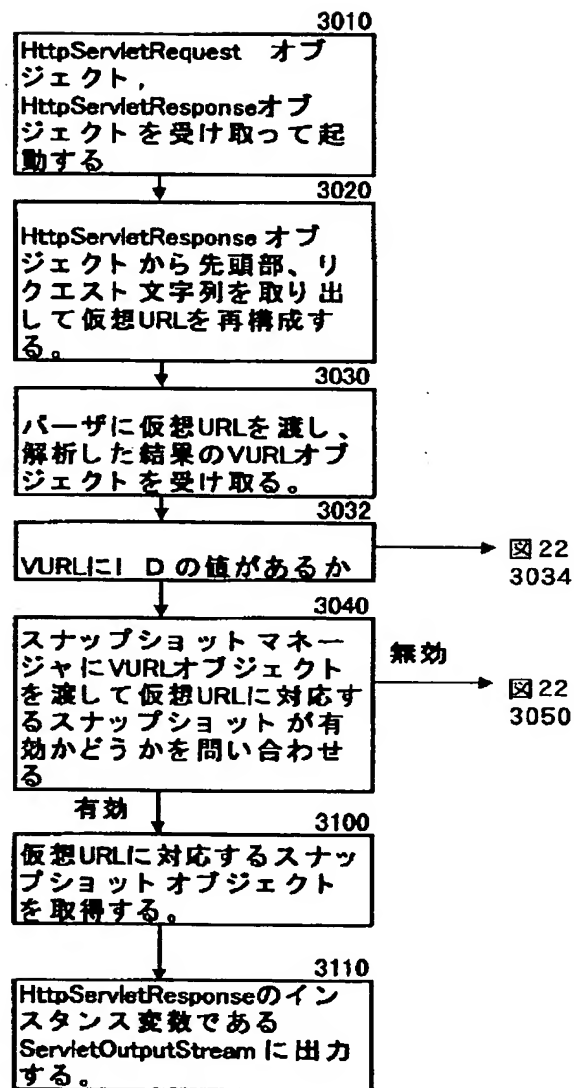
【図19】



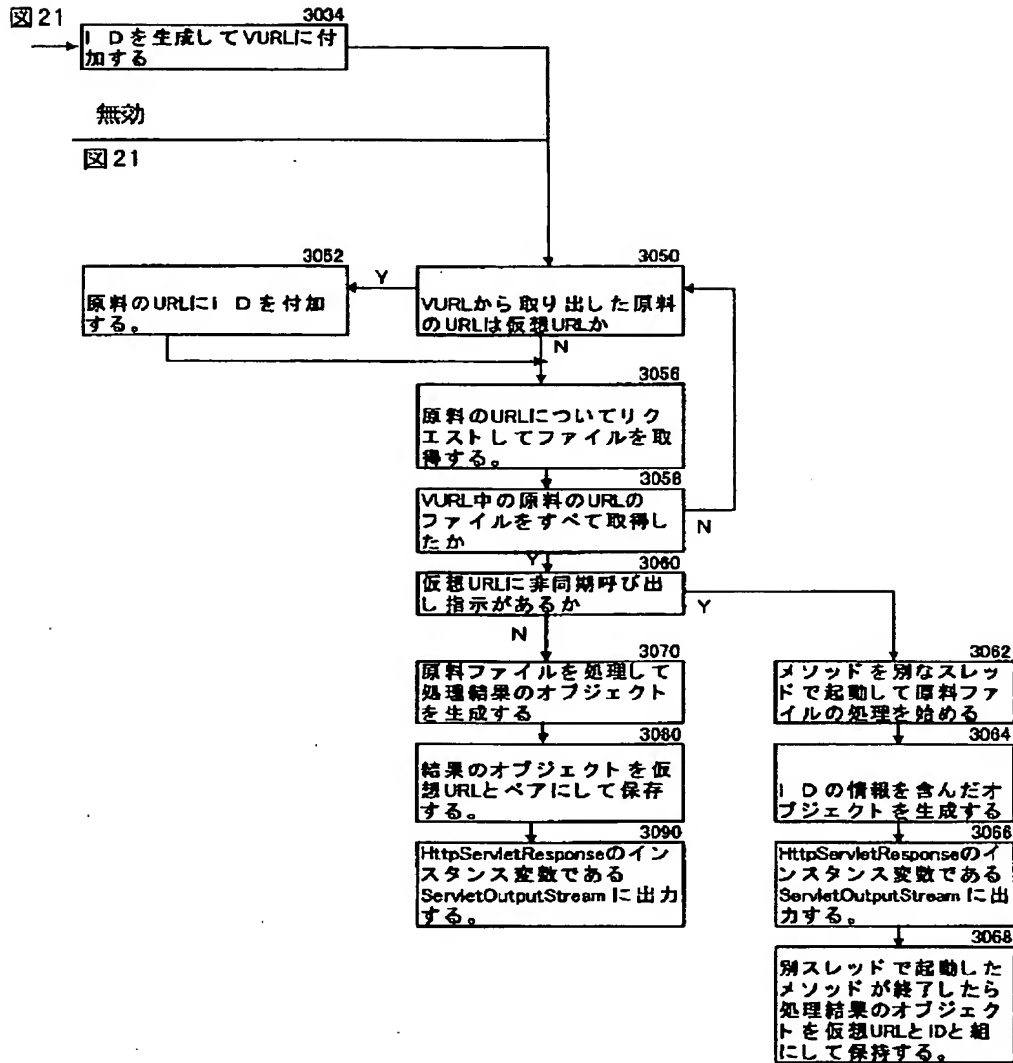
【図20】



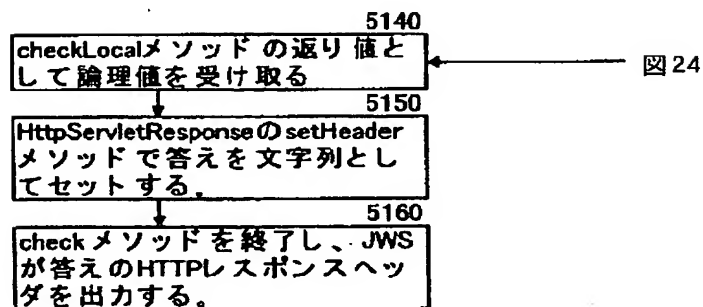
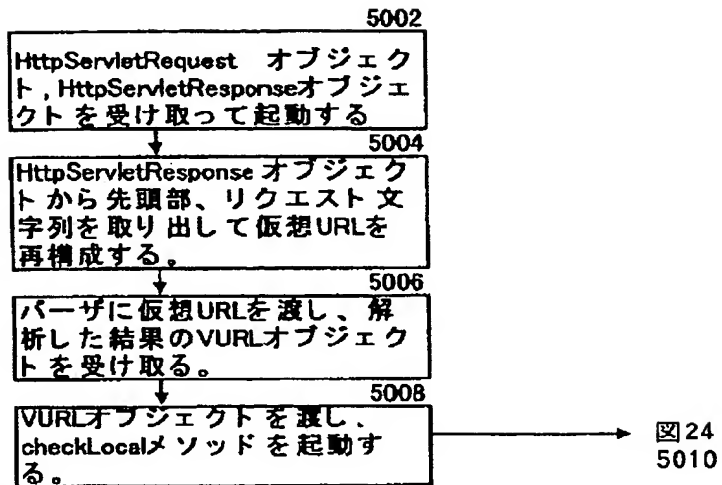
【図21】



【図22】



【図23】



【図24】

